DOI: 10.1111/cgf.15073

COMPUTER GRAPHICS *forum*
Volume 0 (2024), number 0, e15073

Evaluating Graph Layout Algorithms: A Systematic Review of Methods and Best Practices

Sara Di Bartolomeo,[1,2] Tarik Crnovrsanin,[1] David Saffo,[1] Eduardo Puerta,[1] Connor Wilson[1] and Cody Dunne[1]

[1]Northeastern University, Boston, MA, USA
{dibartolomeo.s, crnovrsanin.t, saffo.d, puerta.e, wilson.conn, c.dunne}@northeastern.edu
[2]Universität Konstanz, Konstanz, Germany

Abstract

Evaluations—encompassing computational evaluations, benchmarks and user studies—are essential tools for validating the performance and applicability of graph and network layout algorithms (also known as graph drawing). These evaluations not only offer significant insights into an algorithm's performance and capabilities, but also assist the reader in determining if the algorithm is suitable for a specific purpose, such as handling graphs with a high volume of nodes or dense graphs. Unfortunately, there is no standard approach for evaluating layout algorithms. Prior work holds a 'Wild West' of diverse benchmark datasets and data characteristics, as well as varied evaluation metrics and ways to report results. It is often difficult to compare layout algorithms without first implementing them and then running your own evaluation. In this systematic review, we delve into the myriad of methodologies employed to conduct evaluations—the utilized techniques, reported outcomes and the pros and cons of choosing one approach over another. Our examination extends beyond computational evaluations, encompassing user-centric evaluations, thus presenting a comprehensive understanding of algorithm validation. This systematic review—and its accompanying website—guides readers through evaluation types, the types of results reported, and the available benchmark datasets and their data characteristics. Our objective is to provide a valuable resource for readers to understand and effectively apply various evaluation methods for graph layout algorithms. A free copy of this paper and all supplemental material is available at osf.io, and the categorized papers are accessible on our website at https://visdunneright.github.io/gd-comp-eval/.

Keywords: visualization, graph drawing, graph layout algorithms, evaluation

CCS Concepts: •Human-centred computing → Graph drawings; Visualization design and evaluation methods

1. Introduction

Graph and network layout algorithms map nodes and edges in a graph to coordinates in space—a.k.a., graph drawing (GD). Several aesthetic criteria have been developed to evaluate the quality of the output of a graph layout algorithm. These criteria are usually based on factors that influence the readability of the resulting visualization. For example, readability can be dramatically improved by reducing edge crossings [Pur97]. Graph layout algorithms strive to optimize the placement of the elements of a graph according to such criteria, sometimes directly but often indirectly. Aesthetic criteria are often described by quantitative measures, such as the number of crossings or the total edge length [Pur02, DRSM15].

The process of presenting a layout algorithm often involves demonstrating its performance characteristics through experiments with diverse datasets and a comparison of quantitative results. These experiments, or *computational evaluations*, entail running a set of layout algorithms on a series of graphs and reporting quantitative measures about the resulting layouts and/or the execution of the algorithms. This practice can be referred to as *benchmarks*, a concept that other fields of computer science also utilize. For instance, computer graphics techniques are evaluated on their quality and performance in rendering intricate 3D scenes through graphical benchmarks. Similarly, benchmarks are used in the testing and comparison of electronic hardware, such as the latency of a router.

User evaluations and user studies also play an instrumental role in understanding how a layout algorithm performs in real-world scenarios. These evaluations can reveal aspects such as an algorithm's usability, intuitiveness, and overall user experience, providing a complementary perspective to the more technically oriented

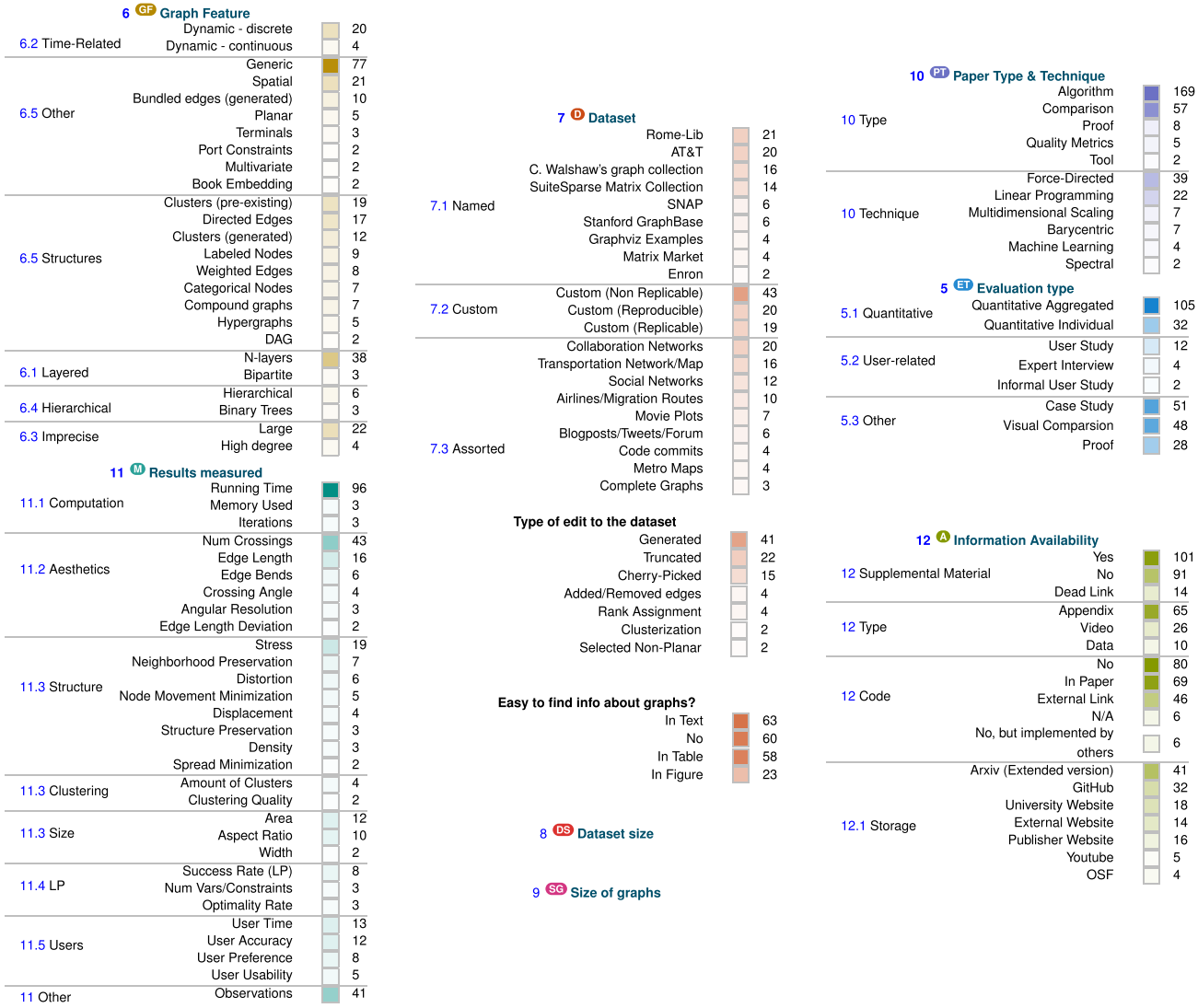


Figure 1: *Ranked lists showing the categories and frequency of features in our collection. The number to the right of each item shows how many papers it was present in. Categories with fewer than two occurrences have been left out. Each paper was tagged with multiple categories if appropriate. Our systematic review can be navigated from this table by clicking on the section number near the titles.*

computational evaluations. They offer valuable insights into how end-users will likely receive and use the algorithm, which is an aspect that purely computational evaluations may not fully capture.

Assessing the results from both computational and user evaluations can guide a visualization designer in making an appropriate algorithmic choice for their specific data and target user tasks. For instance, the questions 'Can this algorithm lay out a graph with 10,000 nodes within 3 seconds?' or 'Which of these two algorithms produces fewer edge crossings for graphs with many high-degree nodes?' can quickly filter out many algorithms. Similarly, user studies can answer questions about the ease of use and understandability of the layouts produced by these algorithms.

However, as this systematic review will show, the community uses a wide variety of computational evaluation approaches. In particular, there is a diversity of benchmark datasets (graphs) used with distinct characteristics, various evaluation metrics are proposed, and results are reported in many different ways. These specifics affect what we learn from a computational experiment and, naturally, what a reader can learn from the results about whether a proposed algorithm fits their use case. The wide variety in computational evaluations makes it difficult to perform meta-analyses and compare results from multiple studies, which is further complicated by missing supplemental materials.

Our contribution: To aid researchers in designing evaluations for graph layout algorithms, we performed a systematic review of 206 papers that present graph layout algorithms. We classified each paper by which graph features the algorithm handles, the evaluation approach used, what datasets were used in the evaluation, the

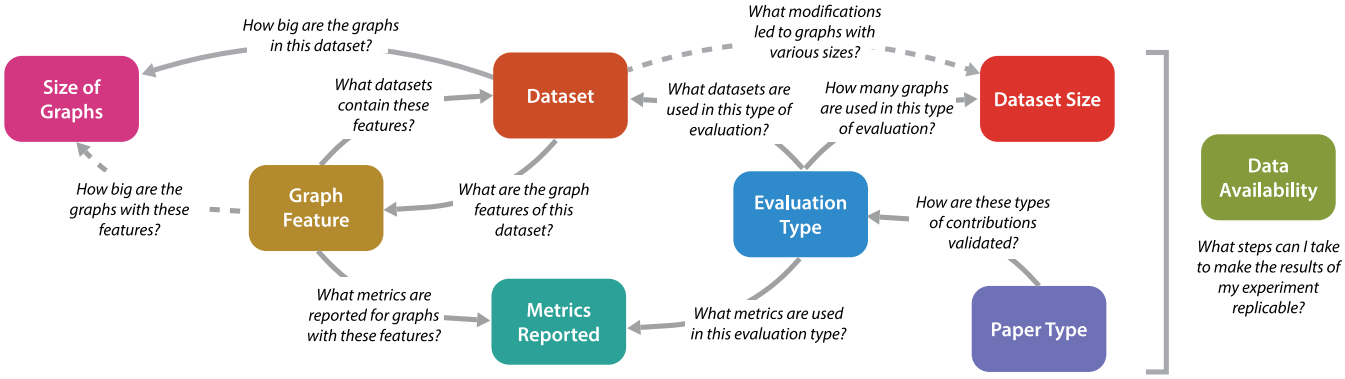


Figure 2: *There are several design choices to make before running a computational evaluation. Here we show elements to consider in the design (boxes) and how each choice informs your other options (arrows). For instance, an algorithm designed to work with* GF *large graphs is likely to be evaluated based on its scalability, which, in turn, leads to more likelihood of reporting* M *running time results. Dashed arrows indicate partial dependencies in which elements are only dependent in some cases.*

number and size of the graphs in the datasets, what metrics were reported and whether supplemental materials were available. We are particularly interested in how these categories influence each other. For instance, an algorithm designed to lay out large graphs will likely be evaluated for scalability, e.g. using run time as a metric on a dataset with large graphs.

The collection of papers we examined and classified is available on the website for this paper at https://visdunneright.github.io/gd-comp-eval/. To ensure that the survey stays relevant, the website lets users add new papers by filling out a form and submitting it as a GitHub issue. We also provide the data as a JSON file and the source code of our website at osf.io [DBCSD23].

2. How to use this paper

Designing an evaluation for a graph layout algorithm implies navigating each element in Figure 2—the choice of evaluation type, choice of metrics to report, dataset size *etc.*—respecting the implications of each choice made. Each of these aspects is extensively detailed in Section 4 which explains our classifications, the implications of design choices and an analysis of how these aspects often interact in published evaluations. This section is not meant to be read sequentially as a narrative. Rather, it is designed so that researchers can quickly find information relevant to their design choices. Figure 1 and the list at the beginning of Section 4 provide an overview of eight aspects to consider when designing the evaluation. Each aspect is assigned a colour (■ ■ ■ ■ ■ ■ ■ ■) so that references to it in the text or chart can be more readily identified.

We also include a series of co-occurrence matrices, which are meant to illustrate how these aspects interact in published evaluations and, in particular, how a choice in one aspect informs their decision space. For instance, a researcher working with *N*-layered graphs can use Figure 10 to look up other GF graph features these graphs commonly have. The top row there shows that these types of graphs often contain dynamic–discrete elements. Likewise, Figure 11 would help the researcher find datasets containing *N*-layered graphs (perhaps Movie Plots) and Figure 8 to look up what metrics are commonly reported with *N*-layered graphs.

We will describe how we designed one of our computational evaluations to illustrate how a researcher can use this paper. Di Bartolomeo *et al.* [DBPB*22] were tasked with representing a network of authors collaborating on VIS papers. Thus, the starting point was the D dataset. As papers may have three or more authors, any co-authorship relationship could logically be modelled as a hyperedge—making the dataset have GF hypergraph features. The fact that the dataset was given also determined the SG size of the graph. After we developed several alternative layout algorithms for these hypergraphs, we chose to write a PT comparative evaluation paper. We used a ET Quantitative Aggregated evaluation, which necessitated running the layouts against a DS large set of graphs. We also created two case studies as an additional ET evaluation type. Finally, we chose which M results to measure: number of crossings, edge length and running time in aggregated form for the quantitative evaluation; individual for the case studies.

Another way you might use this paper is if a peer reviewer asks you to add a user study before accepting your manuscript. Herein you may be able to find M results you can measure in a computational evaluation that are reasonable proxies for the human performance the reviewer cares about—indeed, many aesthetic criteria (section 11.2) are grounded in user studies.

3. Methodology

This section describes the methodology for our systematic review, including how we gathered relevant papers in multiple phases, removed non-relevant papers and validated our categorizations.

3.1. Collecting papers

Our paper collection and analysis process is illustrated in Figure 3.

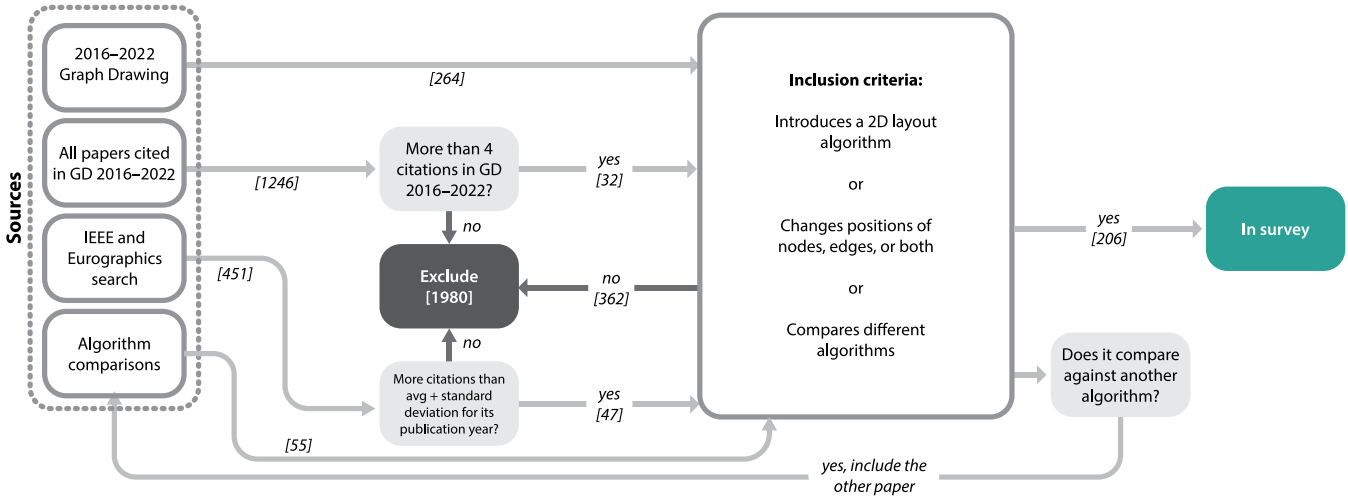


Figure 3: *Literature sources organized in a flowchart. The numbers on the edges represent how many papers were taken into account. We started from 264 papers published at graph drawing between 2016 and 2022, then took into account all of their citations, and added papers collected from IEEE Xplore's and the Eurographics digital library's search functions. Additionally, if the results of an algorithm were compared against something else, we included the compared algorithm as well. After applying our inclusion criteria and filtering out the duplicates, we ended up with 206 included papers.*

Graph layout literature is diverse and widely distributed across many different venues. To aid us in finding all relevant work, we divided the paper collection into two phases. We started with the 264 papers published at the GD Symposium over the last 7 years (2016–2022). The last 7 years of GD provided a good starting point to find a good overview of the most up-to-date research on graph layout algorithms, and additional important work from previous years was included through snowballing and using search engines, as described later.

We manually examined each candidate paper, skimming the content but paying particular attention to the abstract, figures and evaluation approach. We included every paper that introduced a 2D layout algorithm, presented sections of layout algorithms (such as rank assignment or edge bundling) or compares different pre-existing layout algorithms. We consider a layout algorithm to be any algorithm that changes or defines the coordinates of nodes, edges or both. Theorems, proofs, problem statements, posters and challenge submissions were omitted. We still include papers that contain proof as a form of validation as long as the paper itself is about a layout algorithm. In addition, we chose to exclude 3D layout algorithms, as their readability criteria and layout methods are vastly different than 2D ones. For a similar reason, we excluded adjacency matrix visualizations. The selection phase left us with 68 layout algorithm papers from the last 6 years of GD.

GD is not the only venue where layout algorithms and their evaluations are presented. In phase two, we expanded our search to additional venues to paint a more complete picture. We started with the 68 selected GD papers and examined any papers they cited. The objective was to collect the most popular and highly cited layout algorithm papers that were either not published in GD or were published before 2016. Out of 1246 citations, we restricted our list to those with four or more citations from our subset, resulting in 32 candidate papers. Every time a paper had a journal version, we chose to report on the journal version (e.g. CFG for EuroVIS and TVCG for VIS). Duplicates were removed manually.

To include results from other relevant venues (such as TVCG and VIS), we performed an automated search on IEEE Xplore and on the Eurographics Digital Library. We did not search the ACM Digital Library as we are unaware of any predominantly visualization or GD venue with proceedings published by ACM. However, we encourage readers to extend this survey with any relevant papers we have missed. See the Contribution Instructions on our website for details. IEEE's and Eurographics' search engines returned 451 papers that had (1) any word in any metadata from this list: [graph, network, layout] AND (2) filtering for categories [graph theory, computational geometry, optimization, neural nets, trees (mathematics), data structures, computational complexity]. Of these 451 papers, we took into account the number of citations and computed, for every year, the average number of citations and their standard deviation. We restricted our list to papers cited at least the average plus standard deviation for that year. Afterward, we applied the inclusion and exclusion criteria we used in phase one, resulting in 47 candidate papers. This brought the current total to 206 papers. The venues where they were published are shown in Table 1.

Finally, we examined the citations of any of the 106 included papers that compared multiple layout algorithms. The 55 algorithm papers that were cited and fit our criteria went directly into the accept pile. After removing duplicates, this last round of papers brings our total to 206 included papers.

3.2. Validation

For consistency, each paper was examined by at least two authors during all collection and analysis phases. We examined the papers in two cycles. In the first cycle, we started with a base set of

Table 1: *Venues where the papers in this review were published. Whenever a paper had both a journal and a conference version, we chose to report on the journal version.*

Venue	Number of papers
Graph Drawing	94
TVCG (incl. VIS)	40
CGF (incl. EuroVis)	28
PacificVIS	3
Other	41
Total	206

categories we were interested in and expanded from there as we noticed trends. For instance, we noted the presence or absence of supplemental materials from the beginning but later decided to expand our analysis, e.g. including the type of storage used. After completing the first cycle, we cleaned up the collected data, removed redundant categories and codified properties in each category.

In the second cycle, we checked over all the papers again, standardizing the tags used across all papers. The results of our collection and analysis can be accessed on https://visdunneright.github.io/gd-comp-eval/ , where authors can also add new entries or suggest modifications. Our data and supplemental material are also available on osf.io.

4. In-Depth Categorization of Surveyed Papers

In this section, we discuss the process and results of our categorization. Our categorization encompasses how researchers validate their algorithms in the literature. For each category, we will give a formal definition, detail how it was coded and finally proved an in-depth analysis of its results. Some elements of the visual language (such as Figure 1) we use are inspired by Kucher *et al.* [KPK18].

- 5 ET Evaluation Type describes how the authors validate their contributions. Whether the authors chose to evaluate their algorithms using benchmarks on many graphs or examining a few case studies, the evaluation type influences many of the other choices that will be made.
- 6 GF Graph Features captures the characteristics present in the graph that need to be taken into account by a layout algorithm. For instance, the layering in a layered graph is a feature that needs to be addressed, or the presence of clusters of nodes that must be kept adjacent.
- 7 D Dataset refers to the source of data (graphs) used to demonstrate the effectiveness of a layout algorithm. There is a tendency to utilize the same data set as previous similar work so as to support comparison. When available data are insufficient, authors often generate synthetic data—with varying degrees of reproducibility. We followed different procedures depending on the origin of the data. If the data came from a previous collection, we traced the source of the data. If the authors generate the data, we examined how reproducible the process is by reading through method descriptions and looking for supplemental material.
- 8 DS Dataset Size covers the number of the graphs used across the paper. We examined the text of any evaluation section and searched tables or figures with dataset information.
- 9 SG Size of Graphs captures the nodes' range in each paper. We grabbed the lowest and highest node count reported in tables or shown in figures. When insufficient details were present, we estimated the graph size by manually counting nodes in the figures or finding the data source. In both instances, illustrations and toy examples are ignored from the count.
- 10 PT Paper Type and Technique defines the structure and expectations of the paper. It is usually linked to the type of evaluation the paper might conduct. For instance, papers proposing new algorithms are usually expected to compare performance with previous methods.
- 11 M Results Measured are attributes compared between two or more methods. Similar to the Evaluation Type, we traverse through the evaluation and the accompanying figures and tables to derive the information.
- 12 A Information Availability covers multiple aspects of how reproducible the evaluation is, such as the availability of supplemental material, what code is provided (if any), the ease of finding necessary information like data source and graph size and the reliability of any external links. Our ability to compare and verify existing algorithms is intrinsically connected to how easy it is to access the information needed to reproduce prior work. We first checked if supplemental material and source code are present or linked to in the paper, then on a publisher's site, and finally, we did a Google search for the paper's title. Ease of access was checked alongside the Dataset and Size of Graphs collection processes.

5. ET Evaluation Type

> What do I want to communicate to my reader about the algorithm I am presenting? How many instances should I use in my dataset to properly report my findings?

There are many varied ways to communicate computational findings and analytical outcomes. Each method has its unique set of benefits and potential drawbacks that necessitate careful evaluation prior to making a decision on the most effective manner of data exposition. This decision is especially crucial when revealing the inner workings of an algorithm and the resultant computations.

Consider, for instance, the seminal work of Fruchterman and Reingold [FR91a]. Their paper, which introduces an early incarnation of the force-directed layout algorithm, places a significant emphasis on the visual representation of the algorithm's results. By opting for a visual narrative, they engage the reader's intuition and perceptual cognition, offering an immediate and impactful display of the algorithm's output. However, the trade-off of this approach is that it may sometimes eschew the granularity and precision offered by numerical or statistical data.

In stark contrast to this approach, Chimani *et al.* [CMB08] adopted a rather different strategy. They present an aggregate analysis of results sourced from multiple algorithm runs executed on a comprehensive set of graphs. This method effectively captures

overarching trends and recurrent patterns. Moreover, it helps highlight the robustness of the algorithm's performance across a wide variety of graph structures. But one might argue that this approach, while providing a statistical overview, might lack the illustrative clarity that visual diagrams can deliver.

At times, researchers choose to blend multiple presentation methodologies to leverage the unique advantages of each. A case in point is the work of Buchheim *et al.* [BCE*08]. This mixed-methods approach allows for a more comprehensive and nuanced understanding of the data, marrying the illustrative power of visual representations with the statistical rigour of aggregated data. It aims to provide the reader with both the bird's-eye view of algorithmic performance and the specific details of individual results, thus offering a more rounded perspective.

In summary, the choice of how to report the results of an algorithmic operation is a matter of balancing visual clarity, statistical rigour and the intended audience's requirements. The diversity of these approaches reflects the complexity of conveying information in an accessible, informative and engaging manner. This choice heavily influences all the other categories we discuss below, and the evaluation type should be picked carefully while considering what it is that the authors want to communicate to their readers.

Focusing on presenting a limited set of graphs allows for the presentation of in-depth details about each one of the graphs, possibly taking into account and discussing edge cases or particular graphs with special features. A limited number of graphs can also be shown visually, allowing the reader to form an idea about the result of the application of the algorithm at a glance, enabling them to quickly understand if an algorithm could be fit for a specific use case. Conversely, taking into account a large number of graphs can instead offer insights into how the performance of the algorithm changes when used with graphs with different variables: a classic example is showing how the running time of an algorithm increases with graphs of increasing sizes.

Among the papers collected in this survey, we found that most papers used Quantitative Aggregated evaluations (80 papers, as shown in Figure 4), followed by Visual Comparisons (67 papers) and Case Studies (62). As previously mentioned, one of the determining factors in distinguishing evaluation types is the number of graphs taken into account in the evaluation, thus the DS dataset size is going to be vastly influenced by the choice of evaluation type (as highlighted in Figure 6). We also found a relation between the choice of evaluation and the SG size of the graphs used: evaluations with a limited number of graphs tended to use larger graphs, often using them as individual examples to show or compare the visual results of their algorithms on a few large instances. Conversely, aggregate evaluations often used large collections of smaller graphs, without many outliers in the number of nodes. The M quality metrics reported, too, change based on the evaluation type. Figure 8 shows the co-occurrences between the two. It is easy to see how user-related metrics go hand-in-hand with user studies and expert interviews (Figure 4), but it is also interesting to notice that Quantitative Aggregated evaluations are the ones that most often use running time and crossing number as a metric—as a result of wanting to produce visualizations that use numerical values such as the one in Figure 5.

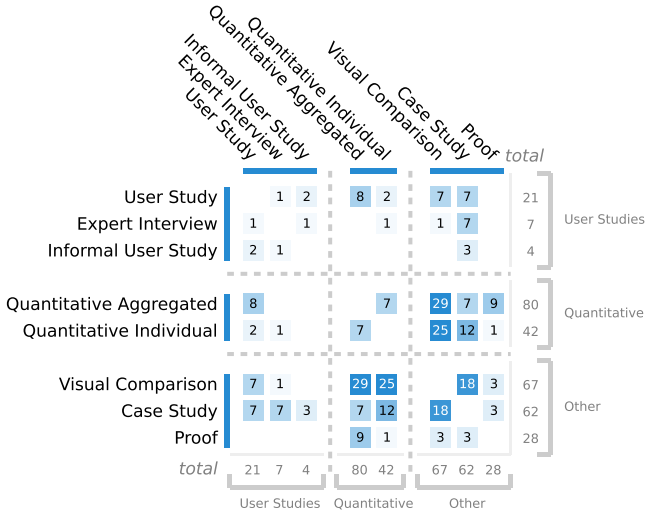


Figure 4: ET × ET *Co-occurrence matrix of what evaluation types are often used together. E.g. a Visual Comparison or Case Study generally complements any Quantitative study. Unfortunately, proofs and user studies were never used in the same paper.*

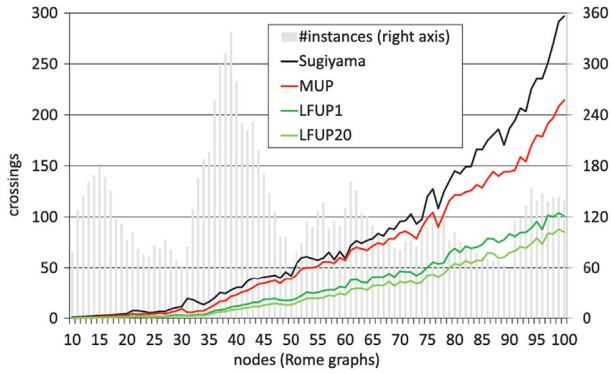


Figure 5: *This figure from Chimani* et al. *[CGMW10] shows a Quantitative Aggregate* ET *evaluation type for comparing multiple algorithms on the same dataset. The goal of the chart is to present the performance (here, the number of edge crossings) of each algorithm with increasing numbers of nodes. At the same time, it displays the distribution of node count for the graphs in Rome-Lib. This is an example of reporting dataset information and comparative results simultaneously.*

5.1. ET Quantitative individual and aggregated

Quantitative individual: In these types of evaluations, quantitative values on a number of relevant metrics are reported for individual graphs. As each graph taken into account will require at least one line in a table or individual discussion, these types of evaluations usually do not take into account more than DS 100 graphs (see Figure 6). Often this is done to show results on specific instances that might contain edge cases or to compare results between two methods in detail, or when the dataset chosen is limited [GSM11].

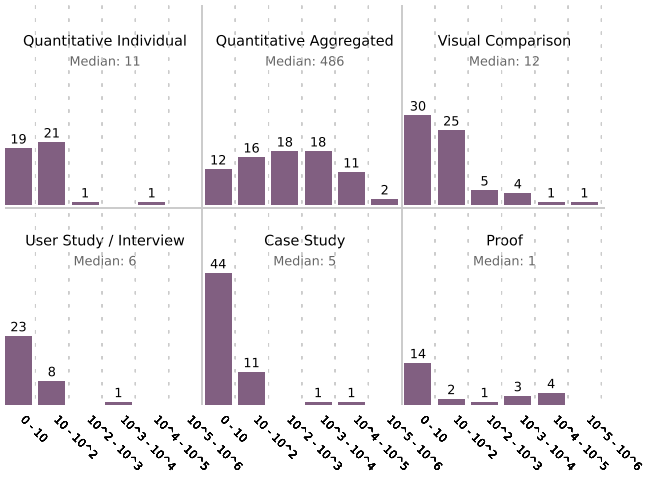


Figure 6: *The chart describes the distribution of* DS *dataset sizes used papers with at least one* ET *type of evaluation. Papers with user studies, case studies or expert interviews usually had the fewest graphs (at most 10). Conversely, quantitative aggregated studies had many more graphs, with a few using up to a million.*

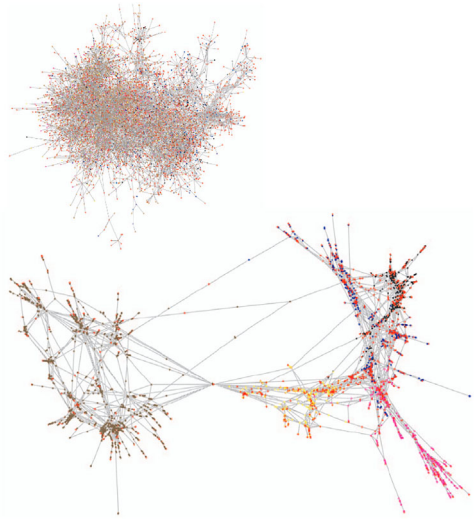


Figure 7: *An example Visual Comparison* ET *evaluation type from Frishman* et al. *[FT07] (bottom). The authors compared their algorithm (bottom) against GRIP [GK00] (top). They chose to use this rendering to demonstrate the efficacy of their algorithm for a particularly complicated graph. The rendering effectively shows a clear difference, although a single example could be cherry-picked to work in favour of the authors' method. To supplement this evaluation, Frishman* et al. *also presented aggregated results.*

These results are often reported on a table where each row is dedicated to a single instance—such as in Ref. [OKB17].

Quantitative aggregated: Here, summarized averages of a number of metrics are reported for a large set of instances of test graphs. For example, the average running time or the number of crossings are very common metrics to report with this evaluation. This is the most common style of evaluation in the papers included in this survey, with 80 papers using it. It allows for summary charts to be reported showing trends—for example, it can be shown how the running time changes with graphs of growing sizes [DBRGD22]. Quantitative Aggregated allows for DS large numbers of graphs to be included in the evaluation, and in the papers included in this survey, it was the category with the most instances included in the experiment, with papers reporting tests on up to a million graphs [GLA*21]. Although this style of evaluation does little to quickly communicate information about the layout's visual output, it allows a reader to gauge how the presented algorithm would perform on graphs with, for instance, a given number of nodes. As the number of graphs taken into account is often very large, individual results are not given for every single graph. Instead, authors report aggregated metrics, such as the total average, or the median result, for a given set of graphs. Well curated examples of this type of evaluation can be found in Refs. [CMB08, OKB17, BCE*08, BFG*18, HJ05b].

5.2. ET User-related

A standard user study can be performed on the results of the algorithm, often collecting both qualitative (such as user preference, or usability) and quantitative metrics (such as accuracy and time to complete tasks). While this is a widespread practice in fields related to human–computer interaction, in the context of layout algorithms running a user study can help test the results of an algorithm when in the presence of specialized tasks, or when the qualities of the algorithm would be hard to describe through standard quantitative metrics.

Similar to the previously described user studies, running an expert interview can help to identify hard-to-classify metrics using specialized experts. For example, Huang *et al.* [HZM*16] present an algorithm specialized for urban networks, and interviews an urban transportation expert to test the usefulness of their system. Liu *et al.* [LSL*17], which presents an algorithm specialized for visualizing neural networks, used an expert as well to test how well the system worked within the reasoning structure of someone who was already used to the context. Both of these evaluation types go hand in hand with M user-related metrics described in Section 11.5.

5.3. ET Other

Visual comparison: The focus of the evaluation is given to presenting final renderings of the results, and comparing them against renderings produced with other methods (see example in Figure 7). Seeing the final rendering directly grants the reader the ability to very quickly see if the result would be apt to a specific use case. Showing individual pictures requires space, thus this style of evaluation often uses DS fewer graphs in the dataset—in the papers in this survey, papers using this style of evaluation did not report more than 100 instances. Like Quantitative Individual evaluations, when working with a limited set of graphs, it is important to avoid cherry-picking the graphs used to work in favour of the authors' method. Visual Comparisons are indeed often accompanied by other

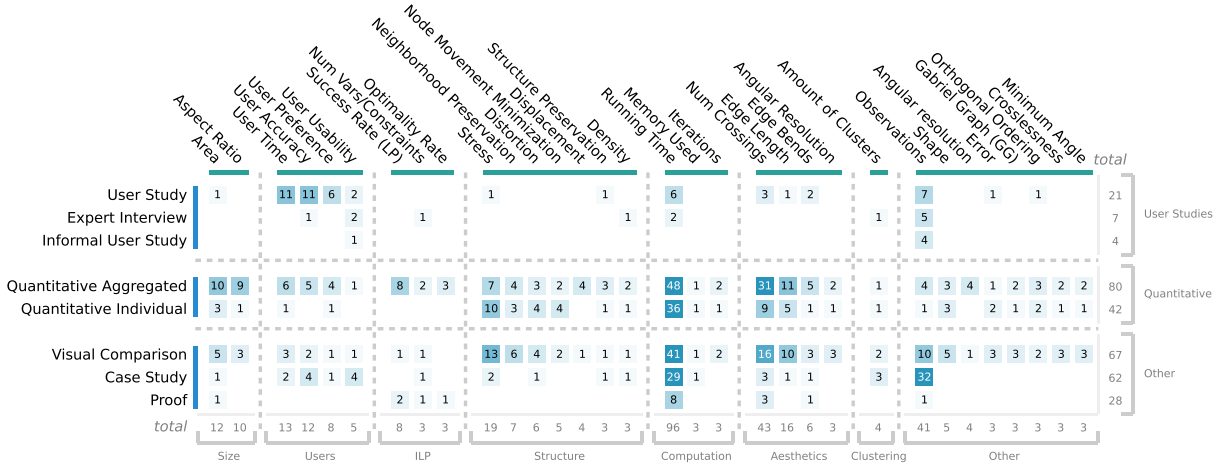


Figure 8: ET × M *What metrics are reported with different evaluation types? It is easy to see, for example, that user-related metrics naturally go well with user studies, but it is also interesting to notice how Quantitative Aggregate evaluations are the ones to most often use the number of crossings as a measure.*

evaluation types, such as Quantitative Aggregated (29 papers contain this combination, examples are [MHE20, WAA*22, GHN13]), Quantitative Individual (25 papers, examples: [NOB15, THM15, WWS*18]) or they are part of case studies (18 papers, [LWW*13, EHP*11, HBH18]). Well curated examples of visual comparisons can be found in Refs. [GHN13, HJ05a, MM08b, TM12, HET*19].

Case study: The algorithm was developed with a specific motivating case study in mind, or the authors want to show how apt the algorithm is for that particular case. For example, Ref. [HFM07] is focused on visualizing social networks, or Ref. [BW98] is instead focused on train interconnection data. Case studies are often accompanied by the author's M observations on how fitting the algorithm is for the particular case (as in Figure 8, where it is shown that the most common way of reporting results in case studies is through the authors' own observations), and they can also include their own specialized task analysis that helps to describe criteria for the algorithm to respect and optimize for.

Proof: Sometimes, authors can use mathematical steps to prove that a method can achieve a certain objective or perform in a certain way. The use of proof as an evaluation technique is more prevalent in papers published in GD. Through proofs, authors can prove without graphs that, for instance, a method can provide results with an optimal number of crossings. Proofs are often accompanied by other evaluation types (most commonly quantitative aggregated with eight papers [dCKN19, NR20, CDMP18], or visual comparisons [Kor05, NNB*17, OT18], or case studies [Ber00, Kor05, BK02]).

5.4. A brief discussion on which evaluation type to use

Quantitative Aggregate and Quantitative Individual evaluation types are effective methods to report information about a layout algorithm. While both provide valuable insights, practicality can guide the choice. When no metric can fully encapsulate the intricacies of an algorithm, though, a user study may be more fitting. In such situations, user accuracy and user time are frequently reported (as illustrated in Figure 8), and they serve as a good basis for reporting. In cases where users cannot be recruited due to the task's complexity, expert interviews can serve as an alternative, potentially supplemented by other evaluation types to demonstrate effectiveness—see, for instance, Refs. [CKS*16, DI18, LSL*17, EASG*17]. In our review, we found 31 of the papers included user studies and expert interviews.

A proof can be a potent tool for showcasing the significance of a result, and in our analysis, they tend to be the sole evaluation type used in papers featuring them [FKR21, KKRS21, BLNN21]. We encourage readers to explore relevant works on our website for insights into crafting these types of papers.

Visual comparisons and case studies can effectively supplement other evaluation types. There was a period when a case study alone was enough to demonstrate an algorithm's capabilities (as seen in Figure 18d). From our review, the preferred number of case studies seems to be three (Figure 6). However, as time passed, reviewers gradually moved away from relying solely on case studies and began favouring multiple evaluation types.

Evaluation methods such as visual comparison, case studies, and to a certain degree, Quantitative Individual, may be subject to cherry-picking biases. The reviewer may interpret these as instances where the algorithm performs at its best or, conversely, does not function in other scenarios, regardless of the author's intent. Similar issues can arise with quantitative studies. For instance, a method may reduce edge crossings but result in longer edges than another method. Without a visual comparison or case study presenting the layouts themselves, assessing the less tangible aspects that contribute to a good layout becomes challenging. Therefore, it is crucial to present both types of information: those with measurable results and those that can only be captured through visual representation.

Figure 9: GF × M *A co-occurrence matrix showing what graph features have been most commonly reported with what results were measured. Results and features that appeared fewer than three times have been omitted. Running time and number of crossings were the most commonly reported measures, as well as observations (discussed in Section 11).*

In summary, **our recommendation** is to employ either Quantitative Aggregated or Quantitative Individual evaluations and supplement the results with a case study or visual comparison. If a metric cannot adequately capture the nuances of the work, consider supplementing a quantitative experiment with a user study.

6. GF Graph Features

> What graph features does my algorithm handle, and how do they affect the design of my computational evaluation?

Graph features indicate intrinsic properties of the graph or visualization that need to be reflected in the layout algorithm. Intrinsic properties include having an assigned layering or hierarchy of nodes, a dynamic graph with timestamped changes, or labels that must be visualized. Special requirements for the layout could include placing all nodes on a line or curve or limiting edges incident to a node to a specific angle.

Graph features is a key category of this survey as it is closely tied to several other choices you must make in your evaluation design. For example, when deciding which M results to report, various properties will require assessment using distinct metrics, thus Figure 9 showing GF × M can be used to check which metrics are most commonly reported for the features you have. Graph features should also be reflected in the D dataset used for testing, thus Figure 11 GF × D can help you identify datasets that have specific intrinsic properties of interest, such as needing datasets with temporal features when designing a dynamic layout algorithm.

A few notes about the words used to describe these categories:

- The word *generic* is used in this context to describe a graph with no specific requirements.
- In case of ambiguity, we always preferred the author's words in their paper. So, in the case of features such as *large*—that can widely vary based on context and date of publication—we tagged the graph as such if the authors' own wordings defined their data using the specific word.

The most commonly addressed graph category was generic (65), followed by *N*-layered (33) and large (19). The following subsections describe in detail several macro-categories and detail their correlation with other elements of the evaluation process.

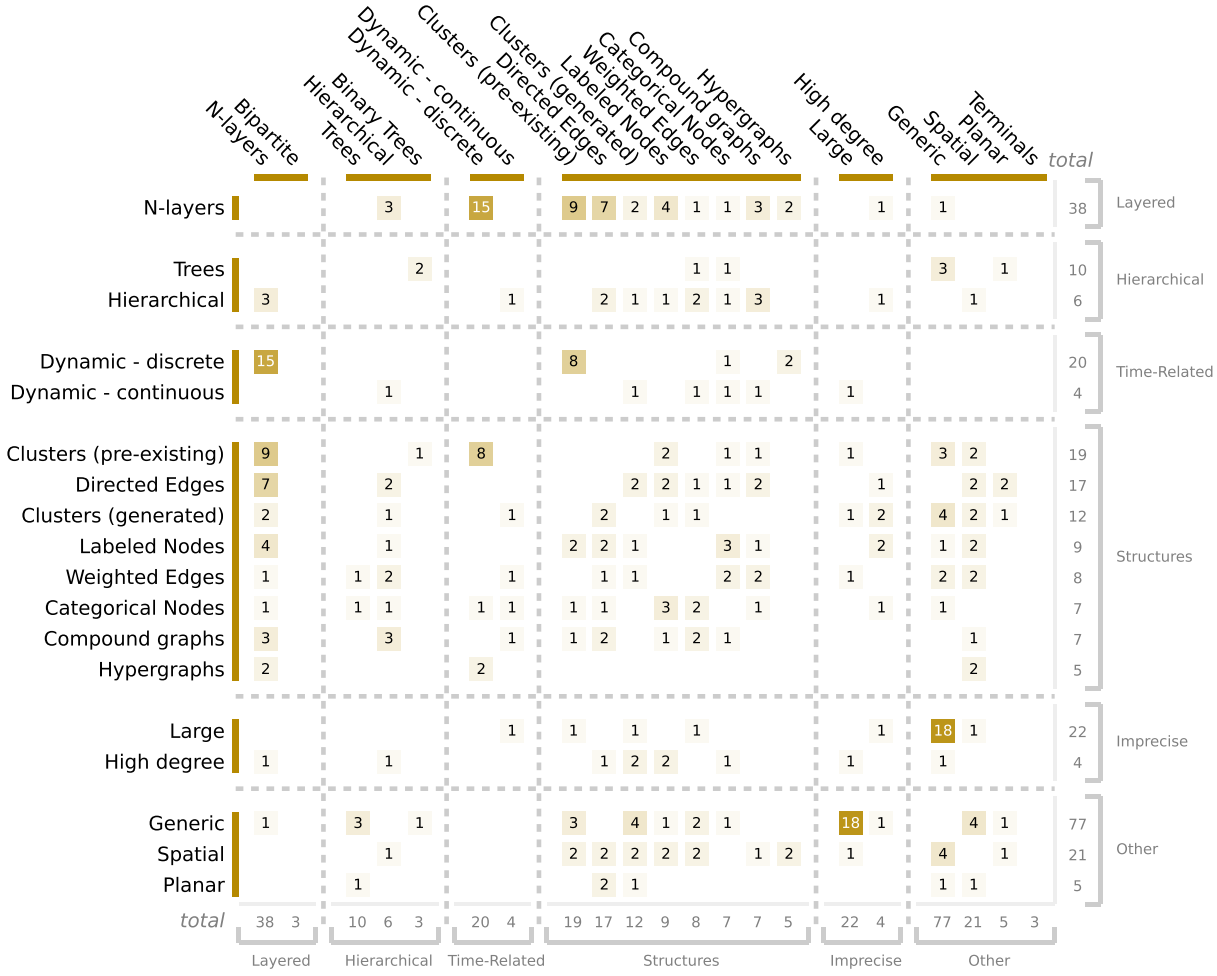


Figure 10: GF × GF *Co-occurrences of different graph features. Categories with fewer than three instances in the collected papers have been omitted. The colouring of the squares is normalized over rows and capped at 10: anything above 10 will have the same colour—indicating that many papers have that combination of elements.*

6.1. GF Layered graphs

In layered graphs, every node is assigned to a layer, and nodes in the same layer are drawn aligned. In some instances, layers can be referred to as *ranks*. The categorization *N-layers* is used when the layered graph contains more than one layer. These types of graphs require specialized algorithms, which might include a rank/layer assignment step, in which nodes are assigned to layers, followed by sorting within the layers.

From Figure 10, we can see that often *N*-layer graphs are used in combination with time-related features (Section 6.2), especially dynamic–discrete, as layers are often used to represent timesteps. This is the case for many storyline-style visualization techniques. This structure of D data is indeed common in narrative plots: many papers dealing with *N*-layers have used the movie plots dataset from Yuzuru Tanahashi to present their results [vDLMW18, dBZSD21, DBRGD22, THM15, GJLM16, TM12, LWW*13]. Other authors expanded on using stories and movies by generating additional similar datasets [PBH18, PBH19]. Collaboration networks that evolve through time, IMDB and internet conversations also map well to *N*-layered graphs [DPF16]. SQL queries can also be represented on layered graphs, using the nesting of the queries to represent layers [DBRGD22]. Another common context is code commits [THM15, OM10, BVB*11] or layers of neural networks [WSW*18, LSL*17]. In cases where a dataset without explicit layering was used, authors used a rank assignment step to define the layering [vdEHBvW13, CGMW10, RESvH16]. Indeed, since one of the most common datasets used for layered graphs has been lost (the previously mentioned movie plots by Tanahashi), some authors have been performing a rank assignment over either the AT&T dataset [JMM*16, JMS18, Mal19] or Rome-Lib [DBRGD22].

In terms of choice of what M metrics to report, layered graphs are not particularly noteworthy and often report the most common metrics such as running time and the number of crossings, sometimes including edge length [DBPB*22, THM15, DBRGD22,

JMM*16, JMS18, BGL*00] or edge bends [WZBW20, LWW*13, BGL*00]. Area and aspect ratio are not particularly concerning when drawing layered graphs due to the already constraining nature of layers. Still, some authors may be concerned about reporting on them as well [JMM*16, RESvH16, BGL*00].

Bipartite is the class of graphs where nodes are split into two groups, and these two groups can be represented as two different layers. While bipartite graphs are not necessarily layered, we included them in this category as a number of papers treat them as such [JM96, BJM02, For05]. For D datasets, researchers have either custom generated them [For05], assigned ranks to non-bipartite datasets [BJM02] or used the individual sample included in Stanford Graphbase, random_bigraph [JM96].

6.2. GF Time-related

Graphs can contain temporal elements. Time can be either continuous (such as timestamped events) or a sequence of events. We distinguish between dynamic–discrete (which are discussed within Layered graphs in Section 6.1)) and dynamic–continuous.

Sometimes, continuous time can be sliced and made into discrete time, according to need [SAK18]. To avoid slicing, continuous time can be represented through animation [GAM14, LSCL10]. Examples of datasets used with what can be considered continuous time are tweets [SAK18, GAM14], or publications in dblp [LSCL10].

6.3. GF Imprecise graph features

Large is a vague word that can have different meanings based on context and year of publication: Figure 14 shows that there are vast disparities in the graphs defined as 'large' by different authors. Nevertheless, defining an algorithm as targeted to large graphs immediately communicates that the algorithm is designed with scalability in mind, and this is reflected in the fact that running time is reported as a metric in 84.2% of the papers in this survey tagged as 'large' (as opposed to 53% when considering the full set). The smallest graphs defined as 'large' we found are in Ref. [FLM95] and contain up to 256 nodes, while the largest graph is found in Ref. [KCH02] and is claimed to have 7,533,224 nodes, taken from an unfortunately now lost George Karypis' collection. Due to the size, papers dealing with large graphs rarely deal with reporting the number of crossings, which is often just not a focus of scalable algorithms, opting instead to report (in addition to running time) on total edge length [ADLM19, FLM95] or stress [MST*14, OKB17].

Datasets for large graphs are often social networks such as the ones contained in SNAP [ADLM19], the Network Data Repository [ADLM19, ENH18] and very commonly C. Walshaw's graph collection [HJ05a, KCH02, BP06, HJ05a, GKN05a, HJ06, ENH18], or the AT&T graph collection [HJ05a, HJ06, HJ05a]. Some have used maps [MM08b, MM08a, NOB15].

6.4. GF Trees and hierarchical graphs

We distinguished between *hierarchical* graphs and *trees*. The former has a more general meaning, while the latter requires a root source for the graph to be clearly defined. The two categories are exclusive in the survey—hierarchical is classified as anything with a defined hierarchy, not a tree. An example of a hierarchical graph that is not a tree is control flow graphs of software execution, which can have more than one entry point. Wongsuphasawat *et al.* [WSW*18], for instance, use the word hierarchical to describe the graphs they use, which are dataflow diagrams of deep learning models from TensorFlow. In some cases, hierarchical can also be synonymous with compound [GBD09] if the hierarchy is defined between sections of the graph.

6.5. GF Structures and other features

Graphs can include a few structural features that layout algorithms must consider. For example, if you want to visualize node labels, the layout algorithm often includes avoiding overlaps on nodes that might have varying sizes.

The applications of graphs with *directed edges* are multiple and varied. In particular, several layout algorithm papers discuss *DAGs*—directed graphs which are also acyclic—as they can be used to affect the layout. For example, the direction of edges can be used to define a rank assignment in layered graphs [JMM*16]. Although there exist collections of DAGs (such as the NorthDAG collection), some authors used a custom dataset [BPWv18] or edited another one to remove cycles [JMS18]. As another property of the edges, *weighted edges* might be interesting when the algorithm deals with forces that can be influenced by their weights [OKB17, GKN05b]—thus can be associated with reporting a measure of stress as a metric.

Clusters are groups of nodes that relate to each other, and often the objective of a layout algorithm that handles clusters is to draw nodes belonging to the same clusters adjacent. Clusters can be pre-existing as part of the dataset: they can be people working in the same context in a collaboration network [OM10, GDL*20] or characters appearing together in the same scene of a movie [vDLMW18, LWW*13, PBH18, PBH19]. Alternatively, if the clusters are not already defined in the data (in our figures, we called this *Clusters (generated)*), the objective of a layout algorithm can be to find and visually highlight groups of nodes that could be grouped. This is the case for social networks, for instance, where the objective could be to find close-knit social circles [Noa04, RW18, QZZ22, MHEK19]. In this case, authors of papers might want to report the number of clusters generated as a metric [LSCL10]. Expanding on the concept of grouping and clustering, the word *compound* indicates a graph that includes subgraphs. Subgraphs can also have their own subgraphs. There are not many publicly available datasets that include compound graphs, so authors have edited the property into pre-existing datasets through clustering or summarization techniques such as Refs. [NRS08, DBRGD22].

Hypegraphs are graphs in which edges can connect more than two nodes. Layout algorithms for hypergraphs have been explored much less than those for generic graphs, but some real-life cases do exist that need to be represented through hypergraphs, such as the VIS collaboration network, where collaboration with *N* authors can be seen as an N-way edge [DBPB*22]. Hypergraphs have several representation styles, and evaluating readability metrics highly depends on the chosen representation style [Mä90]. Because

hyperedges can be represented as groups of nodes (see the set standard representation described in the previously mentioned [Mä90]), the problem of hypergraph representation can have some overlaps with drawing groups or clusters, as shown in Ref. [vDLMW18].

We use the word *spatial* to describe graphs in which nodes have positioning boundaries given by the context. This is often the case when dealing with maps, where nodes must be close to their geographical position. Thus, it is common to test layout algorithms with spatial features on maps [NSM*19], airline or migration routes [HvW09, vLBR*16, EHP*11, WZZY18, WAA*22] or other transportation maps [GBD09, HZM*16, BW98, MRS*13, NW11].

A few papers in the survey present *edge bundling* techniques. Per our inclusion criteria, edge routing is part of a layout algorithm; thus, edge bundling techniques fit within the criteria. Bundled edges are not an inherent property of a graph but can be considered a property of the final drawing and a requirement for the layout algorithm to be taken into account. Papers presenting edge bundling techniques often use high-density datasets where nodes have fixed positions (often spatial) and exclusively work on the routing of the edges [WAA*22, WZZY18, CZQ*08], such as airline or migration routes. Another instance of a high-density dataset is the neural networks used in Ref. [LSL*17]. In terms of metrics, as the objective of edge bundling is to produce cleaner drawings with less clutter, Wallinger *et al.* [WAA*22] include a measure of ink reduction in their reported metrics and report calculated values to measure ambiguity and distortion that might be introduced by the bundling techniques as well [NEH13]. Similarly, Wu *et al.* [WZZY18] include a measure of clutter, also discussed in Ref. [LHT17].

7. D Dataset

> What dataset should I use to test this layout algorithm? What dataset contains features I want to test?

The selection of the dataset for generating results is tightly intertwined with the graph layout algorithm's addressed features. For instance, if an algorithm is designed to manage layered graphs, it is advisable that each graph in the dataset chosen for the experiment has an assigned layering. This consideration is a critical element of our review as we intend to answer fundamental questions about the optimal dataset selection.

A resource we provide is Figure 11, which acts as a guide to help identify the most suitable dataset for addressing a specific graph feature. For instance, if an author devises a layout algorithm for *large* graphs, the figure shows which datasets have been previously used to test other large graph layout algorithms. This can help authors find new datasets to consider. Another resource is Figure 12, which presents, for each graph feature, a list of datasets that can contain it, and a reference to every paper in our analysis that uses the dataset.

An interesting approach to reporting dataset features is observed in Ortmann *et al.* [OKB17]. In their work, each graph used is presented alongside a sparkling depicting the distribution of node degrees within each graph. This approach immediately links the results to the graph's density (or sparsity), inclusive of instances where graphs may simultaneously exhibit both dense and sparse sections, as illustrated in Figure 13. This becomes particularly relevant when the algorithm's complexity correlates with the nodes' degree (as discussed in Section 11.6).

In the next subsections, we have grouped datasets into three categories: 'Named', 'Synthetic', and 'Assorted'. Given the vast number of options, the following sections primarily focus on describing the properties of the most prominent collections or dataset types.

We would like to draw attention to the prevalence of synthetic, custom or edited datasets, which could indicate a potential gap in the availability of datasets encapsulating specific features. This gap could lead researchers to generate their own datasets due to difficulty locating datasets naturally endowed with the features they seek. Creating and popularizing datasets incorporating these absent features could be a valuable contribution to the field.

As a final, noteworthy enhancement, we also provide a collection of easy to navigate benchmark datasets [DBPW*23], which include many of the statistics that, from this survey, appeared relevant to report about a dataset, already computed. The website is accessible at https://visdunneright.github.io/gd_benchmark_sets/.

Recommendation: In terms of data reporting, multiple facets should be taken into account. Essential information about the dataset can be effectively reported in three ways: tabulated, as paragraphs, and in figures that utilize them. Tables are ideal as they allow easy location and extraction of specific information. At a minimum, the table should include the dataset's name, the number of edges, and number of nodes. We also recommend including small sparklines of the dataset's properties and any other relevant information. These visualizations can effectively illustrate the graphs' distribution and convey the data's variability. Furthermore, if the utilized dataset contains graphs exhibiting varied features, these features should be catalogued and reported. Examples of such comprehensive reporting can be found in Kruiger *et al.* [KRM*17] and Chimani *et al.* [CIW21b].

The information about the data's origin should be detailed enough to enable the exact replication of the experiment. For instance, if entire benchmark datasets are used, providing information on where readers can locate the data would suffice. However, in instances where authors do not utilize all datasets from a benchmark, they should indicate which ones were used and provide a detailed explanation in a data section regarding their selection and exclusion. Moreover, we strongly encourage authors to post their data to reliable open archives to enhance reproducibility. See Section 12 A Information Availability for details.

7.1. D Named datasets

Several popular graph repositories and named datasets appeared multiple times while collecting this information. Listing all existing repositories of data would be impossible, but in the papers we surveyed, we found some reoccurring instances that we would like to describe. The following is a brief list of the most used, accompanied by a short explanation of their usefulness in given contexts.

Rome-Lib [BGL*97] is one of the most popular collections of graphs, comprised of 11,389 graphs, having 10–100 nodes and 9–158 edges. It was first introduced in Ref. [BGL*97], and has been

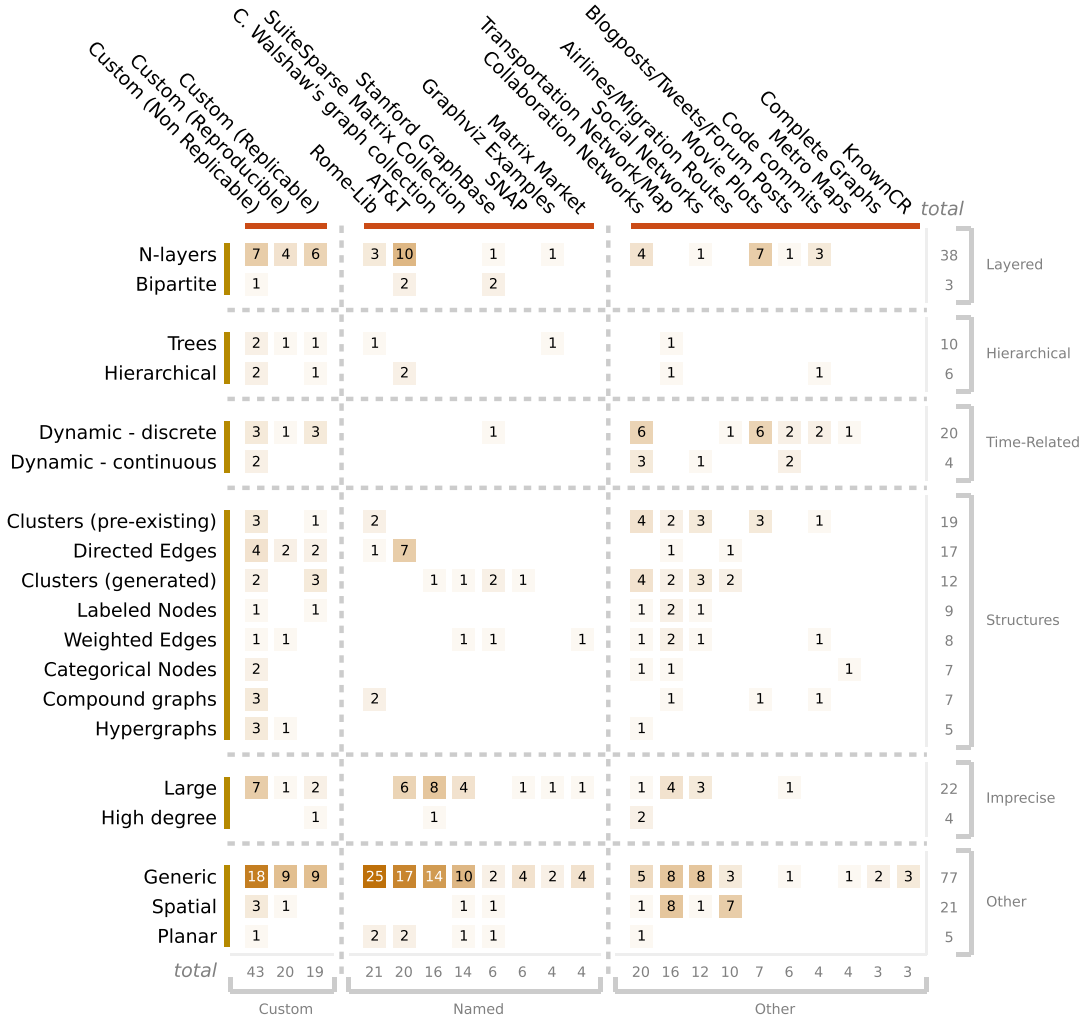


Figure 11: GF × D *Co-occurrence matrix of graph features by datasets used. Features with many instances of custom datasets indicate probable gaps in the available datasets that our community can try to address. For instance, large and* N*-layered graphs seem hard to find.*

used in many layout algorithm papers. According to Ref. [BGL*97], it was originally a collection of natural graphs collected from various sources, such as government organizations and software companies. The somewhat uniform distribution of the graphs in size makes it a great candidate for ET Quantitative Aggregate evaluation types (described in Section 5), as it is used to show on charts how the performance of a layout algorithm grows with increasing SG size of the graphs.

The North DAG (often also called AT&T) collection comprises 5114 graphs with up to 7602 nodes. Per Ref. [BGL*00], 'they are obtained from a collection of directed graphs that North collected at AT&T Bell Labs by running for two years Draw DAG, an e-mail graph drawing service that accepts directed graphs formatted as e-mail messages and returns messages with the corresponding drawings'. Some of the graphs in the collection are not fully connected. Like Rome-Lib, this collection of a large number of graphs is often used in ET aggregate evaluations. Both Rome-Lib and the AT&T collections are available at Ref. [Dra22].

The Graphviz Examples [Gra08] is a collection of varied sample graphs used to demonstrate Graphviz's capabilities. They are few but contain properties that might be of interest for layout algorithms that address particular features [GSM11, NNB*17, CPPS19]. C. Walshaw's graph collection [Wal01b] (online at: [Wal01a]), SuiteSparse Matrix Collection (previously called The University of Florida collection) [DH11] and Matrix Market [NIS07] are all large collections of varied graphs, all related to each other and have vast overlaps between them. They are all very popular and have been used in a plethora of different contexts. Due to the size and variety of graphs in these collections, authors often cherry-pick graphs. Cherry-picking datasets might cause layouts to be tested and presented using graphs that do not include edge cases or particular cases in which the algorithm might not work. Other popular and

Generic: Rome-Lib [HK18, DDM*18, BFG*18, KMP18, GLA*21, CIW21b, CHN19, CG12, GM04, CW16, CGM06, CMB08, BGL*97, BCE*08, BEJ*06] — C. Walshaw's graph collection [HJ05a, Wal01b, FT07, KCH02, Kor05, CMIBR06, Hu05, BP06, HJ05b, GKN05a, HJ05a, ENH18, HJ06] — North DAG - AT&T [HJ05a, HJ05b, HJ05a, BFG*18, HJ06, CG12, DDM*18, KMP18, CIW21b, CW16] — Custom (Reproducible) [FR91b, NLGG18, DDM*18, DAL*19, ALD*20, CIW21b, JVHB14] — Social Networks [Noa04, NOB15, ZJC*21, NLGG18, PAK*20, HJ06] — Transportation Network/Map [Noa04, NOB15, MM08b, MM08a, BW98] — SuiteSparse Matrix Collection [KRM*17, Hu05, GHN13, ENH18, BBP20] — Collaboration Networks [KRM*17, MPL06, GHN13, ZJC*21] — Matrix Market [Kor05, BP06, GKN05b, KMP18] — Airlines/Migration Routes [Noa04, EHP*11, CZQ*08] — KnownCR [CIW21b, CHN19, CG12] — Trade Data [Noa04, NOB15] — Scotch Collection [HK01, KCH02] — GION [MST*14, ENH18] — SNAP [ADLM19, JVHB14] — Complete Graphs [CIW21b, CHN19] — Pajek [Noa04] — Stanford GraphBase [KRM*17] — California [MM08b] — Protein Interactions [MM08a] — Autonomous System Network [ZJC*21] — Internet Mapping Project [GKN05a] — Network Data Repository [ADLM19] — Hachul Library [ENH18] — The Network Repository [ENH18] — FM3 [HK18] — RandDAG [KMP18] — Graph Drawing Contest [DAL*19] — Graphviz Examples [CPPS19] — Planar Graphs [PAK*20] — Blogposts/Tweets/Forum Posts [PAK*20] — Circuitry [CIW21b] — WebCompute [CW16]

Large: C. Walshaw's graph collection [HJ05a, KCH02, BP06, HJ05b, GKN05a, HJ05a, ENH18, HJ06] — North DAG - AT&T [HJ05a, HJ05b, HJ05a, HJ06] — Transportation Network/Map [NOB15, MM08b, MM08a] — Social Networks [NOB15, HJ06] — Scotch Collection [HK01, KCH02] — GION [MST*14, ENH18] — SuiteSparse Matrix Collection [OKB17, ENH18] — Trade Data [NOB15] — California [MM08b] — Protein Interactions [MM08a] — Matrix Market [BP06] — Internet Mapping Project [GKN05a] — Graphviz Examples [NNB*17] — Control Flow graphs [NNB*17] — Custom (Reproducible) [NNB*17] — SNAP [ADLM19] — Network Data Repository [ADLM19] — Hachul Library [ENH18] — The Network Repository [ENH18] — Blogposts/Tweets/Forum Posts [GAM14] — IMDB [GAM14] — Collaboration Networks [GAM14]

High degree: Collaboration Networks [HFM07, ADM*19] — Internet Mapping Project [GKN05a] — C. Walshaw's graph collection [GKN05a] — Neural Networks [WSW*18] — Fiscal Network [ADM*19] — ***Sparse:*** Rome-Lib [KMS18]

Dynamic - discrete: Movie Plots [vDLMW18, dBZSD21, THM15, GJLM16, TM12, LWW*13] — Collaboration Networks [OM10, GDL*20, DPF16, DBPB*22] — Padia stories [PBH18, PBH19] — Enron [vdEHBvW13, THM15] — MID Network data [TM12, LWW*13] — Airlines/Migration Routes [PS20] — Metro Maps [PS20] — IMDB [DPF16] — Blogposts/Tweets/Forum Posts [DPF16] — Protein Interactions [DPF16] — Chess Games [dBZSD21] — Custom (Reproducible) [dBZSD21] — Diabetes data [dBZSD21] — Code commits [THM15] — Stanford GraphBase [GJLM16]

Dynamic - continuous: Collaboration Networks [SAK18, GAM14] — Blogposts/Tweets/Forum Posts [SAK18, GAM14] — Social Networks [SAK18] — DBLP [LSCL10] — IMDB [GAM14]

N-layers: Movie Plots [vDLMW18, dBZSD21, DBRGD22, THM15, GJLM16, TM12, LWW*13] — North DAG - AT&T [JMM*16, JMS18, Mal19, RESvH16, BGL*00, CGMW10] — Custom (Reproducible) [BVB*11, JMM*16, WZBW20, dBZSD21] — Collaboration Networks [OM10, DPF16, DBPB*22] — Rome-Lib [DBRGD22, CGMW10] — Padia stories [PBH18, PBH19] — Code commits [BVB*11, THM15] — Neural Networks [WSW*18, LSL*17] — Enron [vdEHBvW13, THM15] — MID Network data [TM12, LWW*13] — Investment Interdependence [STT81] — DBLP [BVB*11] — Cable Plans [WZBW20] — IMDB [DPF16] — Blogposts/Tweets/Forum Posts [DPF16] — Protein Interactions [DPF16] — Chess Games [dBZSD21] — Diabetes data [dBZSD21] — SQL queries [DBRGD22] — Stanford GraphBase [GJLM16] — Graphviz Examples [GSM11] — World Greenhouse Gas Emissions [ZBD*18] — Pert DAG [BGL*00] — ***Multilevel:*** SuiteSparse Matrix Collection [HET*19] — C. Walshaw's graph collection [HET*19]

Bipartite: Stanford GraphBase [JM96, BJM02] — North DAG - AT&T [BJM02] — ***Tripartite:*** DBLP [LSCL10]

Hierarchical: AT&T [JMS18] — Investment Interdependence [STT81] — Neural Networks [WSW*18] — Code commits [Hol06] — Transportation Network/Map [GBD09]

Trees: Evolution [CDMP18] — Rome-Lib [KMS18] — Custom (Reproducible) [OMK*18] — Graphviz Examples [CPPS19] — ***Binary Trees:*** Evolution [CDMP18]

Compound graphs: Rome-Lib [DBRGD22] — Code commits [Hol06] — Transportation Network/Map [GBD09] — SQL queries [DBRGD22] — Movie Plots [DBRGD22]

Clusters (pre-existing): Movie Plots [vDLMW18, DBRGD22, LWW*13] — Padia stories [PBH18, PBH19] — Collaboration Networks [OM10, GDL*20] — Rome-Lib [DBRGD22] — Medical Patient Records [CGSQ11] — Car Features [CGSQ11] — Biological Pathways [WNV20] — SQL queries [DBRGD22] — MID Network data [LWW*13] — World Greenhouse Gas Emissions [ZBD*18]

Clusters (generated): Social Networks [Noa04, RW18, QZZ22] — Collaboration Networks [KRM*17, HFM07, QZZ22] — Transportation Network/Map [Noa04, MM08b] — Airlines/Migration Routes [Noa04, vLBR*16] — Pajek [Noa04, MHEK19] — Stanford GraphBase [KRM*17, RW18] — Neural Networks [WSW*18, LSL*17] — Trade Data [Noa04] — SuiteSparse Matrix Collection [KRM*17] — C. Walshaw's graph collection [Wal01b] — California [MM08b] — SNAP [MHEK19] — DBLP [LSCL10]

Labeled Nodes: Transportation Network/Map [NW11, HZM*16] — Collaboration Networks [ADM*19, QZZ22] — Neural Networks [WSW*18] — Fiscal Network [ADM*19] — Biological Pathways [WNV20] — Social Networks [QZZ22] — ***Labeled Edges:*** Neural Networks [WSW*18]

Categorical Nodes: Collaboration Networks [ADM*19] — Fiscal Network [ADM*19] — Biological Pathways [WNV20]

Weighted Edges: Matrix Market [GKN05b] — Tobler's flow mapper [BSV11] — DBLP [BVB*11] — Code commits [BVB*11] — Custom (Reproducible) [BVB*11] — SuiteSparse Matrix Collection [OKB17] — Social Networks [RW18] — Stanford GraphBase [RW18] — Transportation Network/Map [GBD09]

Directed Edges: North DAG - AT&T [RESvH16, BCD*16, BGL*00, JMM*16] — Neural Networks [WSW*18, LSL*17] — Custom (Reproducible) [JMM*16, BPWv18] — Transportation Network/Map [GBD09] — Pert DAG [BGL*00] — ***DAG:*** AT&T [JMS18] — Custom (Reproducible) [BPWv18]

Bundled edges (generated): Airlines/Migration Routes [CZQ*08, WZZY18, WAA*22] — Neural Networks [WSW*18, LSL*17] — Code commits [Hol06] — Custom (Reproducible) [WAA*22] — Amazon [WAA*22]

Planar: North DAG - AT&T [CKW16] — Rome-Lib [CKW16] — SuiteSparse Matrix Collection [KRM*17] — Stanford GraphBase [KRM*17] — Collaboration Networks [KRM*17] — SteinLib [CKW16]

Spatial: Transportation Network/Map [MRS*13, NW11, HZM*16, GBD09, BW98] — Airlines/Migration Routes [EHP*11, HvW09, vLBR*16, WZZY18, WAA*22] — SuiteSparse Matrix Collection [KRM*17] — Stanford GraphBase [KRM*17] — Collaboration Networks [KRM*17] — Medical Patient Records [CGSQ11] — Car Features [CGSQ11] — Tobler's flow mapper [BSV11] — World Maps [NSM*19] — Custom (Reproducible) [WAA*22] — Amazon [WAA*22]

Multivariate: Medical Patient Records [CGSQ11] — Car Features [CGSQ11] — Neural Networks [WSW*18]

Hypergraphs: Custom (Reproducible) [LK21] — Collaboration Networks [DBPB*22]

Port Constraints: Cable Plans [WZBW20] — Custom (Reproducible) [WZBW20]

Figure 12: *Every* D *dataset mention, divided by what* GF *graph features were addressed by the algorithm presented in each paper. This table can be used as a reference to answer the question 'my algorithm handles this graph feature—which dataset should I use to evaluate it?'. Following the references, you can see how each dataset was used, and if edits or particular modifications have been taken by the authors.*

graph	n	m	$\delta(G)$	$\Delta(G)$	$D(G)$	$\{deg(i)\}$	$\{d_{ij}\}$
dwt1005	1005	3808	3	26	34		
1138bus	1138	1458	1	17	31		
plat1919	1919	15240	2	18	43		
3elt	4740	13722	3	9	65		
USpowerGrid	4941	6594	1	19	46		
commanche	7920	11880**	3	3	438.00		
LeHavre	11730	15133**	1	7	33800.67		
pesa	11738	33914	2	9	208		
bodyy5	18589	55346	2	8	132		
finance256	20657	71866	1	54	55		
btree (binary tree)	1023*	1022	1	3	18		
qh882	1764*	3354	1	14	32		
lpship04l	2526*	6380	1	84	13		

Figure 13: *This table from Ortmann* et al. *[OKB17] details the* D *dataset used in their computational evaluation. It shows the number of nodes, edges, the min. and max. degree and the diameter, but the last two are of note, showing the* distribution *of degree and distance. Asterisks indicate bipartite graphs.*

varied collections are the Stanford SNAP repository [LS16] and Pajek [BM06]—both offer loose graph descriptions and some information regarding their sizes. These collections are particularly fitting for ET visual comparison evaluation types and quantitative individual evaluations, as usually a limited number of graphs is selected from them to showcase in detail the effect of the algorithm on the particular case.

In some instances, it might be useful to know in advance the optimal number of crossings, especially if trying to compare a heuristic-based method against optimality, as in Ref. [CHN19]. Chimani *et al.* [CIW21b] provide a collection of graphs with a known crossing number, accessible at Ref. [CIW21a], as well as the optimal number of crossings for Rome-Lib.

The Enron dataset [Coh22] is a collection of around 500,000 emails between 150 Enron executives and other employees. Its advantage, when employees are represented as nodes and emails as edges, is the large diversity in node degrees, with nodes that are highly connected and others that are sparsely connected.

7.2. D Generated

Some layout algorithms are designed to handle certain GF graph features that might be more difficult to find in large public datasets. For instance, dynamic graphs can be much harder acquire but are still an important object of study in GD. To address this issue, many authors have either crafted their own dataset programmatically (examples: Refs. [HJ06, ALD*20, HKNN19]) or modified another one to fit their needs (discussed in the next section).

We made a distinction between *custom (non-replicable)*, *custom (replicable)* and *custom (reproducible)*. All three mean that the dataset was custom-made for a specific paper, but the first (*non-replicable*) indicates that the paper contains either too few or no descriptions of the generation process, making the computational evaluation impossible to replicate. The second, *custom (replicable)*, indicates that the generation process is described in enough detail that similar graphs with similar properties could be generated. This is the case when the authors use established generation methods such as Barabási-Albert [BA99] or Erdős-Rényi [ER*60]. The last category, *custom (reproducible)*, indicates instead that the experiment can be repeated using the same graphs. This is mostly the case for papers that distribute their dataset as supplemental material. In some cases, too, there is no ambiguity in how the graphs are generated, such as when authors use complete graphs of increasing sizes [CIW21b, CHN19, dCKN19]. In our survey, we found 68 papers using custom datasets, 25% being replicable, 26.5% being reproducible, while the rest (48.5%) was non-replicable.

Natural versus synthetic datasets: Datasets can be divided into 'synthetic' and 'natural'. The first word describes a dataset that has been programmatically altered, while 'natural' describes a dataset that has been collected from a real use case and underwent no alterations in structure and number of elements (no slicing, no adding features). The two approaches can have different advantages and disadvantages. Using a natural dataset ensures that the algorithm is tested on realistic use cases with realistic purposes. If, for example, a given use case has quirks that might not be known to a researcher developing a layout algorithm, such as a high variance in node degree distribution, this feature might be missed if the dataset used for the computational experiment is synthetically generated. Conversely, synthetically generated data can (a) replace non-existing or private data by simulating it, and (b) create edge cases for testing that might not appear in natural datasets but might still be useful for testing, such as an entire set of complete graphs with varying amounts of nodes.

Similarly to programmatically generating a dataset, some authors edited another pre-existing dataset to have the feature they needed, such as performing a layer assignment on Rome-Lib to obtain a layered dataset (examples: Refs. [CGM06, DBRGD22]). Common edits are listed in Section 1. Editing or generating a synthetic dataset can impact the reproducibility of the experiment. Therefore, either the method used needs to be explained in complete detail, the generated dataset needs to be made accessible, or the code used needs to be distributed (ideally all three).

7.3. D Assorted

Authors sometimes used data from different large sources such as transportation networks and protein interactions. We aggregated the different datasets under the same category in cases that come from similar natural scenarios, which we could assume had similar structures, features and purposes.

Contexts such as a *collaboration network* can be useful for multiple particular properties: collaborations can involve more than one participant, making them a good metaphor for hypergraphs [DBPB*22, QZZ22] or for clusters [GDL*20, OM10]. Additionally, collaborations often occur over periods of time, so they can be useful for dynamic graph visualization as well [BPF14, SAK18], or for a combination of all of these properties. A common collaboration network used often in layout algorithm papers is the VIS collaboration network [IHK*17]. Actors' participation in movies on IMDB can also be used for the same purpose [DPF16, GAM14]. Another context sharing the same properties (being able to be mapped to hypergraphs or clusters, and being dynamic) is movies or book plots. The original movie plots dataset from Tanahashi is

difficult to find (as discussed in Section 12.1), but other authors have started contributing with their own movie plot datasets, such as Ref. [KBI*18].

Blog posts, tweets and forum discussions can offer good hierarchical or directed networks, as multiple threads of conversations can span from a root post or linkage between them can serve as directed edges.

Social networks are good examples when the layout algorithm is trying to highlight non-predefined clusters, such as wanting to identify close-knit groups of friends [Noa04, RW18, NOB15]. One example of a commonly used dataset akin to a social network is Zachary's karate club introduced in Ref. [GN02], the popular Les Miserables dataset found in the SparseMatrix collection.

Transportation networks, maps and airlines or migration routes are especially useful when the layout addresses spatial constraints. When nodes have positioning constraints, often the layout algorithm either focuses on minimizing the distortion compared to the original positioning (13 papers—examples: Refs. [WXW*20, CZQ*08]) or focuses on edge bundling or routing instead of node placement (15 papers—examples: Refs. [WAA*22, ZSJT19]). Due to the fact that they are contained in size and must be represented particularly well, metro maps have been an especially interesting problem for exact layout algorithms [NW11, NR20].

8. DS Dataset Size

How many graphs should I use in my evaluation?

The size of the dataset, referring to the count of unique graph instances used in the evaluation process, is a key aspect of computational analysis. Figure 6 offers guidance on the number of graphs that should be deployed in an evaluation. This largely depends on the chosen ET type of evaluation. For instance, authors tend to use fewer graphs to illustrate their findings in Quantitative Individual evaluations, Case Studies and Visual Comparisons—where the focus is on a meticulous exploration of an algorithm's impact on singular graphs. For these categories, the median lies at or slightly above 10. On the other hand, authors tend to use a much larger collection of graphs in Quantitative Aggregate evaluations that aim to analyse the results of an algorithm on a vast array of graphs with different node counts. In fact, the median for this category rises to 536 graphs per evaluation, based on the papers assessed in this survey. Different graph instances may be created in certain scenarios by sampling subgraphs from a larger graph. Under these circumstances, even if the subgraphs overlap with each other, we classify them as distinct graphs. For instance, if an evaluation utilizes a single graph, such as the Enron dataset, divided into 100 subgraphs, we count it as 100 individual graphs.

The decision regarding the number of graphs to use is influenced by the availability of datasets, but Figure 6 can provide a rough estimate, based on median values and charts, of how many graphs should be employed in an evaluation. Once the authors have settled on the type of evaluation and the metrics to be reported, another pertinent question arises: what is the appropriate number of datasets to utilize? Figure 14b details the distribution of dataset sizes employed for each type of evaluation. For example, the median dataset size in a Quantitative Individual evaluation is 11, while in a Quantitative Aggregate evaluation, the median ascends to 486. The specific count will inevitably depend on data availability.

Authors should consider employing datasets that vary in terms of their source, application and scale. For instance, dissecting a book into chapters and considering each as a separate dataset will not yield the same diversity as using several independent books. The fewer the datasets used, the more diversity is required to effectively showcase the extensive range of algorithm applications.

In light of these insights, our **recommendation** for authors is to use a dataset size comparable or larger to those used in recent related work. Ideally, the data should be derived from a benchmark dataset or gathered from various sources and applications to ensure its diversity. This approach will not only enrich the breadth of the evaluation but also potentially contribute to the reproducibility and generalizability of the research findings.

9. SG Size of Graphs

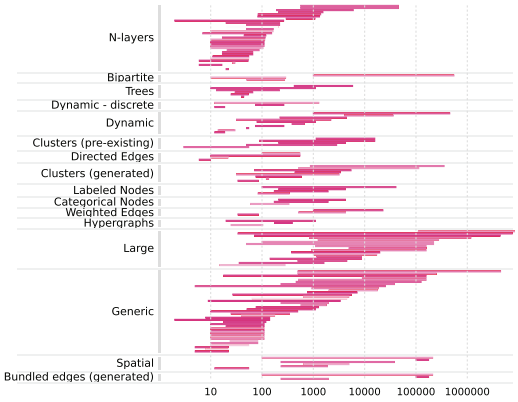
The size of the graphs indicates how many nodes are found in the graphs used in the evaluation. It should be noted that the number of nodes is not the only measure of a graph's size, and the number of edges should also be taken into account. However, we decided to focus on the number of nodes as it is most frequently made explicit and considered when discussing graph size.

The data we collected are stored as a range for each paper, as the dataset used can contain larger and smaller graphs. The size of the graphs used is mostly dependent on the method's target graph features and type of algorithms: if a method is made to handle larger graphs, it will test the results presented on sets or individual graphs that contain large graphs. Figure 14 shows that the size is dependent on the type of algorithm used as well, as some types are more scalable than others, thus able to handle larger graphs—an aspect that is discussed in the next section, Section 10. The same figure also shows that large graphs are more frequent in evaluation types that focus on individual graphs, where a quarter of the papers use graphs with more than 100,000 nodes.

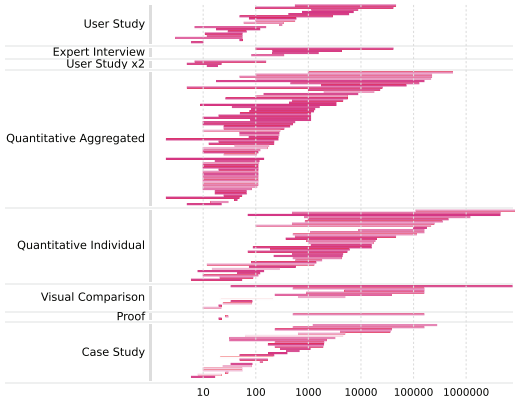
If a specific graph size is required to show the scalability of an algorithm, this should inform the decision on the choice of D dataset.

10. PT Paper Type and Technique

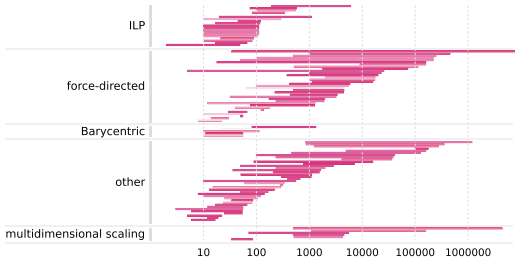
We classified the papers in the survey in 'algorithm', 'comparison' and 'proof'. Indeed, the papers in this survey do not exclusively present algorithms: they might also compare them. This is the case when either (a) the authors introduce a new method and test it against a pre-existing method, or (b) the paper is about comparing different pre-existing methods for a specific objective. In a few of the papers included in this survey, the efficacy of the algorithm is proven through mathematical proofs. In addition, we included a few papers that define or test new quality metrics (such as Ref. [MHE20] or [KMP18]) while still running computational evaluations.



(a) SG *Size of graphs* × GF *graph feature*. *Graphs the authors proclaim as "large" are generally, but not always, the largest.*



(b) SG *Size of graphs* × ET *evaluation type*. *Quantitative Individual and Visual Comparison evaluations are commonly used with largest graphs.*



(c) SG *Size of graphs* × PT *paper type*. *Being more scalable than other types, the class of force-directed methods has been tested on larger graphs.*

Figure 14: *The range in the* SG *sizes of graphs (in terms of nodes) that have particular* GF *graph features,* ET *evaluation types and* PT *techniques. Each horizontal line represents a paper, extending from the size of the smallest graph to the size of the largest one used in its evaluation. Size is shown on a logarithmic scale. Categories with fewer than three papers are omitted. Papers that used a single graph (or multiple graphs with the same node count) are shown with a single dot. Darker lines indicate more recent papers.*

In parallel with the paper type, we classified papers loosely regarding what type of algorithm they used. If they, for example, mentioned their method being an improvement over other force-directed methods, we classified the paper as belonging to force-directed methods. This can be used to inform decisions like the SG size of graphs used in the evaluation (see Figure 14b): ILP-based methods are constrained to smaller graphs due to being more demanding in computational resources, while force-directed and MDS-based methods are usually more scalable, thus have been used with larger graphs in their evaluations. Because of their heaviness of computational resources, ILP-based methods are often more frequently the object of Quantitative Aggregated evaluation types and often require a number of precautions when reporting their results (as described in Section 11.4), while other types of techniques are used with a more diverse set of evaluation types and reported metrics. The *other* category indicates layout algorithms that are not based on other, pre-existing methods or whose roots are unclear.

11. M Results Measured

What metrics are reported for graphs with these features?

Different layout algorithms are optimized for various metrics that must be reported accordingly. Additionally, GF graph features may also require distinct metrics for accurate representation. The primary objective of this categorization is to assist researchers in identifying the results they need to report for a given graph. The metrics to be reported can be influenced by the ET type of evaluation being conducted, evident from the co-occurrences shown in Figure 8. For instance, metrics related to the user experience are reported when user experiments are executed. Likewise, graph features also play a pivotal role in guiding the metrics to be reported, as illustrated in Figure 9. E.g. algorithms that revolve around clustering might report on the number of clusters generated. A visualization of what metrics are reported together can be found in Figure 15.

In situations where authors did not report any quantitative or qualitative metric, but merely provided comments on their results, we classified them as *observations*. Although less objective than other reporting methods, observations can offer insights that might be challenging to express otherwise. Observations are most useful if they are accompanied by other metrics.

Our recommendation: The decision of what information to evaluate can be complex. The metrics employed can vary significantly from one work to another, and hinge on the work's contribution and the explored graph features. For instance, reporting on running time is logical when dealing with large graphs for a quantitative study. However, an algorithm designed to minimize the number of edge crossings would also benefit from reporting running time (along with the number of edge crossings). Reporting multiple metrics becomes crucial when there is a trade-off between a metric and running time. While there is no universal solution, a general guideline when unsure about which metrics to use is to start with those used by the algorithms that the authors intend to compare their work against. There will be cases when authors have to replicate someone else's algorithm. Replication can be necessary when previous authors provided only pseudocode or no information about the

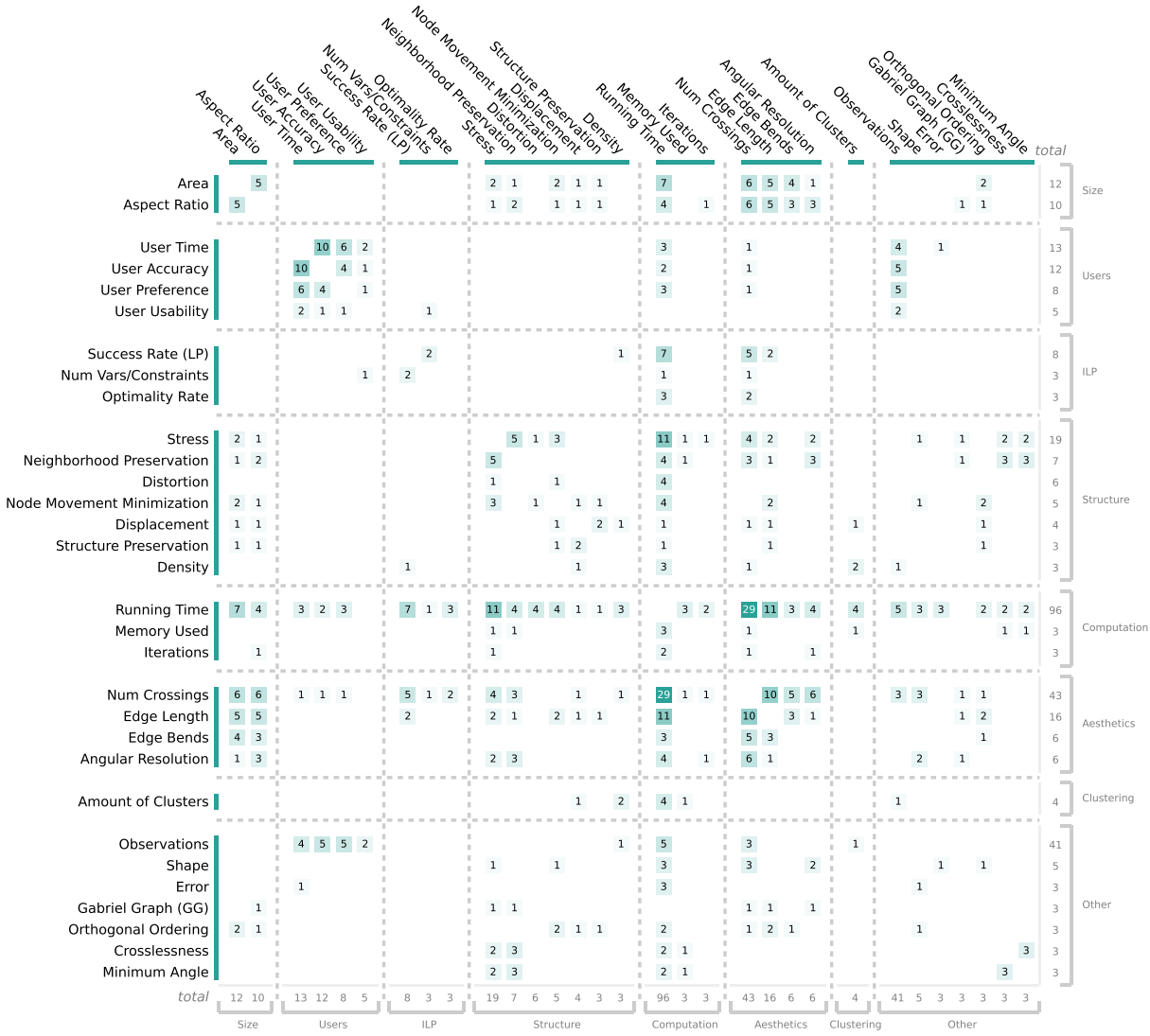


Figure 15: M × M *What metrics are usually reported together? Running time and number of crossings are the most common combination, but we also have a relationship between area, aspect ratio and the number of crossings and edge length. In addition, multiple user-related metrics are often reported together.*

implementation. In such cases, it is essential to provide implementation details either in supplementary materials or directly within the paper. This step aids in offering a more holistic experiment and can assist others who have conducted a similar experiment or are planning to do so.

11.1. M Computational resources

Running time and *memory used* both refer to the requirement of computational resources and not the quality of the result. Running time is by far the most reported metric and was used in 80 of the papers we surveyed. Computational resources are always going to be limited, and reporting running time gives readers an understanding of the algorithm's practical utility. Far fewer papers report the amount of memory required, such as in Ref. [vDLMW18].

While running time can be a great form of comparison between different algorithms within the same paper, it cannot be directly compared across papers because even slightly different machines with slightly different setups or conditions are going to invalidate the results of the experiment. Thus, authors should report at least a few details indicative of the machine used for the experiment. Regardless, reporting running time can (a) still give a loose indication of how long it might take to run an algorithm on a graph of a given size, (b) indicate how the algorithm scales if done on graphs of growing sizes, or allow for comparison between different experiments run on the same machine. Reporting the running time is particularly common in ET Quantitative Aggregated evaluations.

The *number of iterations* can sometimes be reported [BFG*18, Coh97, MM08a] and it can give an idea of how long it takes to

reach convergence or to terminate an execution for algorithms that run in iterations, without having to rely on the running time, which is dependent on hardware and current conditions of the system. Cohen [Coh97] uses iterations as the *x*-axis on a chart to show how their chosen metrics improve progressively less at every iteration. Muelder and Ma [MM08a] integrate iterations in their results of comparing different algorithms to discuss how different methods take longer per iteration, and how many iterations are needed to reach termination.

11.2. M Aesthetics

Many aesthetic criteria can be used to evaluate the readability of a layout [Pur02, DRSM15]. Human-subjects studies have been used to determine which criteria affect user task performance and their relative optimization importance, as well as measures such as user preference. Dunne *et al.* reference many such studies [DRSM15].

According to Purchase [Pur97] (and much follow-up work), the number of crossings is the criterion that most negatively affects readability. This is reflected in the results reported in the papers we reviewed, where the number of crossings is the second-most reported metric (39 papers). It does not have particular dependencies associated with graph features or datasets. In the case of PT non-exact algorithms, sometimes authors might want to use datasets in which the optimal number of crossings is already known, such as Ref. [CIW21a], to be able to compare their results against the optimum. It should be noted that there is a debate around which metric should be used to count the number of crossings, and authors should consider specifying the metric used in their results [PT00].

A few algorithms care about minimizing *edge length* as part of their optimization function—reducing the distance between connected nodes, as long edges can also hinder readability. When the rendering allows for edges to be drawn as splines, minimizing the *bends* in edges can also be important. Such a problem is common in GF layered visualizations that allow bends, where the curviness of the paths may make them harder to follow [LWW*13, WZBW20]. Niedermann [NR20] defines an exact method to minimize bends in orthogonal drawings.

Researchers might also care about angles—from two different points of view: (a) edges incident to a node should be drawn at evenly spaced angles, if possible, to improve how distinguishable they are (defined as *angular resolution* [FHH*93, BST00]), and (b) edges that cross should do so at an ample angle to help following both edges without creating confusion [DL13] (between 70° and 90° according to Ref. [HHE08]).

11.3. M Rendering size and structure preservation

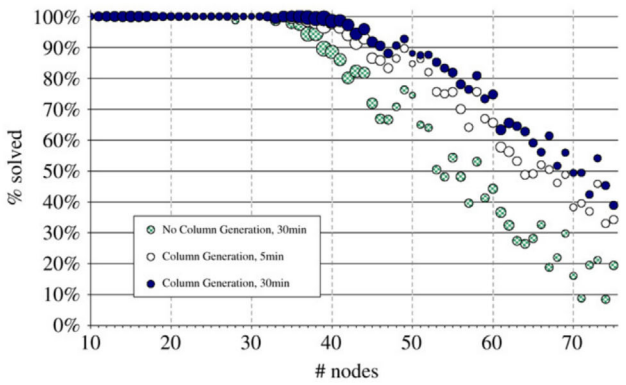
A number of layout algorithms take into account the final size of the rendering and the proportions of the resulting drawing. The *area* needed to represent a drawing is the canvas surface needed to properly represent a graph and can be reported when algorithms take into account GF features that need proper space to be represented, such as node sizes or labels. It is also implied when the nodes are constrained to integer coordinates. It is relevant in the case of PT force-directed/spring-embedder methods, as the repulsive forces used in them might cause drawings to take up more space than needed. Hu *et al.* [Hu05], for example, include diameter in the metrics used to compare different methods. Because a larger drawing area is undesirable due to difficulties navigating the visualization or resolving the marks [DAL*19], restricting the drawing area has been the objective of two GD contests [DGNS13, GLNR14].

The aspect ratio is a key metric in algorithms that adjust the embedding of existing drawings and is used to measure the preservation of the original global shape. Examples of these can be seen in node overlap removal algorithms [LEN05, CPPS19, SSS*12]. The goal is to maintain the overall structure of the graph while eliminating overlaps. Node movement minimization is another metric often used to evaluate the structural preservation of a graph [CPPS19, SAK18]. This metric quantifies how much each node in the graph needs to move to resolve any issues, such as overlaps or crossings. Displacement is yet another metric relevant to structure preservation [CPPS19, NSM*19, FT04]. It measures the total movement of all nodes in the graph during the layout adjustment. Neighbourhood preservation is a measure of how well the graph theoretic neighbours match the realized neighbours (via distance) in the layout [GLA*21, ALD*20, DAL*19, KRM*17]. Finally, distortion is a metric used to assess how much the relative positions of the nodes change after the layout adjustment [WAA*22, SAK18, WZZY18, FT08]. The goal here is to minimize the change in distances between nodes. it is worth noting that some of these metrics can overlap in certain cases. However, we have adhered to the authors' specific terminologies when categorizing these metrics to ensure clarity.

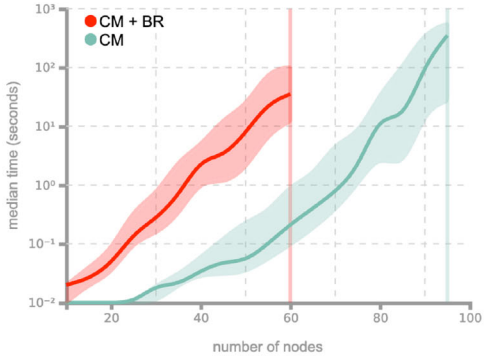
11.4. M Metrics related to linear programming

ILP-based methods often include additional ILP-related metrics that are relevant to be addressed. One example that stands out is reporting the number of variables and constraints being generated during the formulation of the problem [BCE*08, NW11, CGM06], as this can vastly affect the time spent solving an LP problem. Another common metric to be reported by LP paper is the *success rate*, discussed below. When comparing a heuristic against an LP-based method, the *optimality rate* [vDLMW18] has also been included in the results reported (example in Figure 16a).

Reporting running time without termination guarantees: ILP-based methods have exponential growth in computational complexity with respect to the input size. For this reason, computational execution results sometimes include graphs where a computation could not terminate within an allotted time. In the case of *aggregate* reports, this might cause a problem, as having a set of instances that could not terminate might cause issues in computing an average—indeed, the *running time* of a computation that could not terminate is not the upper bound of the allotted time (it could take much more than that), and neither is infinite time. Indeed, a number of papers report the percentage of graphs that could be solved [CGM06, CKW16, CvGM*18, CW16, DBRGD22, CMB08, BCE*08, BEJ*06] within a certain timeframe (Figure 16a). A solution to this is to report the median, and not the average running time, such as in Ref. [DBRGD22], where the charts are interrupted whenever at least 75% of the graphs with a specific number of nodes could not be computed. Reporting the median guarantees that the value is



(a) *Buchheim et al. [BCE*08] used this figure to show how many of their graphs with a given SG number of nodes could not be laid out within the time limit. A very similar solution is used in [CMB08].*



(b) *Di Bartolomeo et al. [DBRGD22] report the median, top quartile, and bottom quartile running time for graphs with increasing numbers of nodes. To ensure correctness, they cut the data when they could no longer lay out 75% of the graphs at that SG size within the time limit.*

Figure 16: M *Results measured: Two examples of dealing with algorithms that do not always terminate in a given amount of time.*

not influenced by the running time of instances that could not terminate, whose time is unknown (it can neither be considered infinite, nor the maximum allotted time)—see Figure 16b.

11.5. M User-related metrics

In some instances, computational evaluations are accompanied by a ET user study to reinforce the claims that the proposed algorithm improves the results [HZM*16, dBZSD21]. Such user studies can include qualitative elements or quantitative elements. In case of quantitative user studies, the authors reported *user accuracy* [MRS*13, CGSQ11, HZM*16, BPF14, KMS18, MA19, PAK*20, DRMM13, QZZ22, dBZSD21] and *user time* [MRS*13, CGSQ11, BPF14, KMS18, MA19, PAK*20, DRMM13, QZZ22, dBZSD21] spent solving a task. When the evaluation included a qualitative element, *usability* [NW11, CGSQ11] and *preference* [BPWv18, KMS18, DRMM13, QZZ22, dBZSD21] were also included. We included in our survey some papers that exclusively have user evaluations: [MRS*13, CGSQ11, BPF14, BPWv18, KMS18, MA19, PAK*20, DRMM13, QZZ22].

11.6. M Reporting complexity

The reported complexity of a layout algorithm expressed in the number of nodes and edges does not necessarily explain the difficulty of solving a problem thoroughly. An example is the barycentric method [STT81], whose complexity is expressed as $O(|N| * |E|)$, where N is the set of nodes, and E the set of edges in a graph. However, this is the worst-case scenario, and the time spent computing a layout will vastly differ based on the average degree of the nodes. For this reason, it would be preferable to include in the description of the dataset information such as the average degree of the nodes in a given graph, or, when possible, use an approach such as the one used by Ortmann *et al.* [OKB17] (Figure 13) where the degree distribution is represented on a chart for each of the graphs they used in their experiment.

12. A Information Availability

A primary objective of this survey is to explore ways to enhance the reproducibility of computational experiments. Thus, the accessibility and availability of supplementary materials, as well as clear information about datasets employed, become crucial.

In the course of our analysis, we evaluated the availability and nature of supplementary resources accompanying the papers. It emerged that a substantial number of papers provided additional materials, often in the form of expanded paper versions or appendices available on arXiv (65 papers, of which 41 are on arXiv)—a trend particularly prominent among papers submitted to GD. However, we also discovered that several links to these supplementary resources have unfortunately become defunct, particularly when they were hosted on university websites or similar platforms requiring regular maintenance, or where the URL was subject to change. The implications and underlying reasons for this are discussed further in Section 12.1.

Another pivotal component when it comes to replicability is the provision of code. Although a paper may present a thorough explanation of the algorithms and methodologies used, providing access to the code or pseudocode can prevent potential misinterpretations. Of the papers in our survey, 69 offered at least pseudocode, while others stored their full code on external repositories. it is noteworthy that code sharing appears to be a more common practice in recent times. Older papers less frequently offer access to their code, but the most impactful among these have seen their code incorporated into tools, thereby enhancing the precision of replication and the overall utility of their research.

The dissemination of supplemental materials has grown increasingly prevalent, supported by publisher websites that permit these materials to be included alongside the main body of the paper. Furthermore, reliable open-access repositories have provided a platform to upload various files. As a result, it is now more straightforward than ever to provide all necessary resources—layout algorithm code, benchmark graphs, analysis code and results—for reviewers

and readers to verify and reproduce the experiments detailed in the paper, and even to build upon the original work.

However, there are circumstances where sharing materials can pose challenges, such as when the code forms part of proprietary software, or when the data employed is owned by a company. We recommend that authors share as many relevant materials as possible and provide a clear rationale for any materials that are omitted. For example, pseudocode can serve as the next-best option if sharing the full code is not viable. Similarly, sharing a synthetic example graph mirroring the properties of a private graph can facilitate code execution and partial validation of results. With substantial storage space available for supplementary materials on reliable open-access repositories, providing readers with necessary information has never been easier.

We also urge reviewers to adopt a more discerning approach to papers lacking essential supplementary materials. Ask yourself—will this paper withstand the scrutiny of future researchers? Would an independent team be able to reproduce or build upon this work based solely on the provided resources? Are there materials that, if included, would expedite and improve these processes?

As a **recommendation**, we advocate for authors to include all supplementary materials necessary for verifying or reproducing the computational experiments they report. This encompasses any code used to create or modify datasets, layout algorithms used, analysis code and the raw results. We further advise that all supplemental material be hosted on reliable open-access repositories such as osf.io, arxiv.org, biorxiv.org, psyarxiv.org and hal.*.fr. Ideally, the existence of supplementary materials and its location should be mentioned at the outset of the paper and referenced throughout the paper whenever relevant to the discussion at hand.

12.1. A Type of storage—Avoiding dataset loss

The disappearance of supplemental material can cause huge issues in instances where other researchers are trying to replicate or extend a work of research. One outstanding example is the movie plots dataset, first introduced in Ref. [TM12] by Yuzuru Tanahashi and crafted manually by collecting which characters appear together in subsequent scenes of popular movies. The dataset was a popular one because it was particularly useful for storyline-style visualization, and temporal event sequences are hard to come across. Several papers cite it as their dataset source [LWW*13, THM15, GJLM16, vDLMW18, dBZSD21]. However, the dataset was stored on Tanahashi's personal page on UC Davis' website, and in the time between 2012 (when it was uploaded) and now, the page is no longer available, resulting in the movie plot dataset being lost. The same dataset is not found in any of the previously mentioned paper's supplemental materials, each one of them referring to the original now-inaccessible webpage as a source. The loss of this dataset caused all the papers using it to become far less reproducible.

In 2018, Haroz [Har18] analysed where supplemental material was stored in VIS papers to discuss the phenomenon of 'link rot', discovering that 5% of papers published at VIS have their supplemental material go missing within 2 months. Vines *et al.* [VAA*14] reports that the availability of a dataset goes down by 17% per year after the year of publication of a paper. The risk for supplemental material to go missing is much higher if it is stored on a personal website or lab webpage, or, even worse, if it is kept on a private storage device and available upon request to the authors. A solution to this problem is to store the material on reliable open-access repositories such as osf.io, arxiv.org, biorxiv.org, psyarxiv.org and hal.*.fr (and more, as listed in Ref. [Har18]). Any chosen repository should enable (1) free access to the data without cost or requiring creating an account, (2) persistent identifiers to avoid link rot and allow citations, (3) post-submission updates for errata, (4) immutable versioning (for pre-registration) and (5) a long-term plan for providing materials in perpetuity. GitHub, like CodePlex before it, would not satisfy (5) as it is owned by Microsoft and subject to corporate pressures.

In this context, it is also worth mentioning that in recent years, several initiatives have been aimed at encouraging reproducible and replicable research. The graphics replicability stamp [Sta] is one of these, meant to be an endorsement of the reproducibility of at least one figure presented in a paper. Authors can opt to have their work scrutinized by another reviewer, who focuses entirely on ensuring that some results can be reproduced. ACM has a more robust badge initiative for open practices [ACM], or the SIGMOD availability and reproducibility initiative [SIG], which goes one step further and publishes full reports on how reproducible a paper is. Finally, Figure 17 describes how graph features inform evaluation types.

13. Changes in Computational Evaluations Through Time

When looking at the papers across time, it is clear that the internet influenced how papers are written. First, starting with A supplemental material, appendices were pretty rare in the 2000s; see Figure 18a. If they existed, they are found at the end of the paper, after the references. As the years went on, supplemental material started to become more common. From 2006 to 2016, more dead links appeared as individuals hosted the material on their own websites or school sites. After 2016, one of the largest shifts was the rise of research-specific sites like arXiv and OSF and the rising popularity of version control sites like Github. The research sites made posting and maintaining files easier, while the version control made it easier to provide code. Around this time, there was also a shift for publishing websites, like IEEE Xplore, to host supplemental material within the website. An interesting trend occurred around 2020, with a large influx of papers containing appendices. GD around 2016 officially recognized arXiv as where all paper will be hosted. Afterward, papers in the GD conference started to reference the full or extended version of the paper on the arXiv site. We are unsure what started this trend, but by 2022, most GD papers had an extended version.

We see a similar internet influence regarding A code availability. Early on, it was common to find pseudocode in papers, as papers were distributed physically; see Figure 18b. If an implementation of the layout existed, it was on popular layout algorithms and implemented by others like Fruchterman & Reingold and Kamada & Kawai. As the popularity of and shift to using the internet for reading and sharing increased, the number of papers containing pseudocode started to decline.

Interestingly, by 2012, the shift was not from pseudocode to external links, but that code was not shared at all. Based on our own

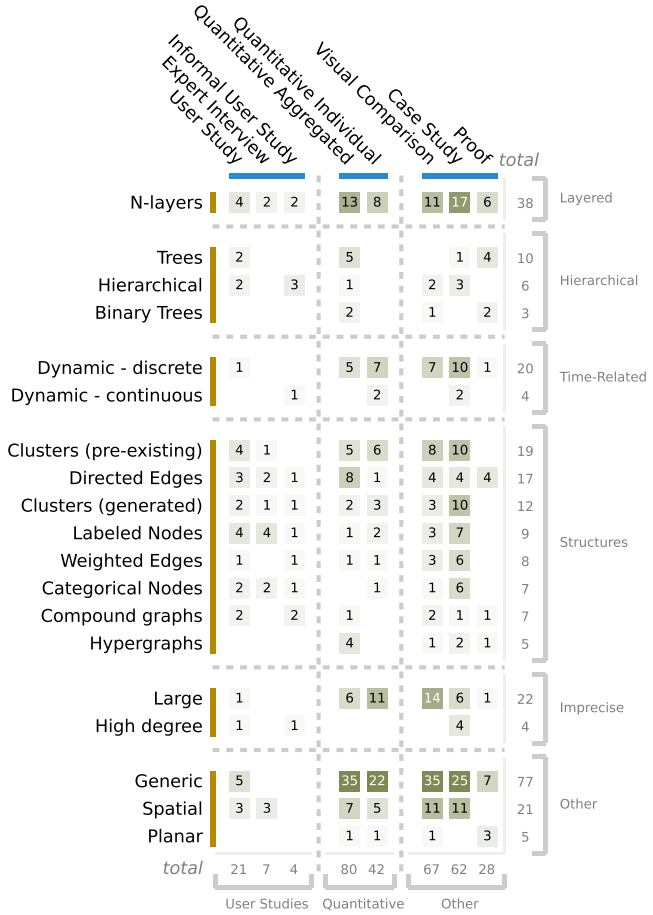
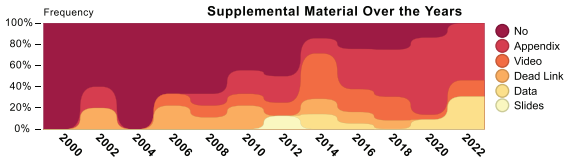


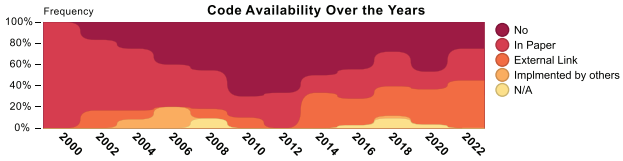
Figure 17: GF × ET *Do graph features inform the appropriate type of evaluation? Some features are widely treated in the context of case studies—such as generated clusters, which are especially common for social network analysis. Others, such as large graphs, are more difficult to test in large quantities with quantitative aggregate evaluations and are tested more with visual comparisons.*

experience publishing papers during this time, we argue that several factors led to this shift. First, reviewers were lenient on needing to provide access to code. Most likely assumed the software and implementation had a good reason for not being shared, such as being proprietary. Second, the field of visualization and, in turn, graph layouts, exploded in popularity. This popularity caused an arms race of sorts, where it was not advantageous to share the code for fear of someone scooping your work. It is important to note that this anecdotal evidence, based on the authors' experience at the time, should be taken as such. Recently, there has been an increase in code being provided on sites like GitHub, though more can still be done in this regard. A larger push for open science and the need for reproducibility could be the driving force.

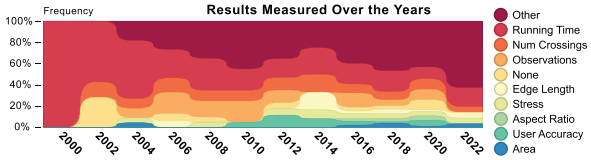
One area that has changed considerably over time is the M results measured. Running time was the most popular metric and one of the few reported (see Figure 18b). Over time, other metrics started to appear as researchers figured out how to quantify a 'good layout'.



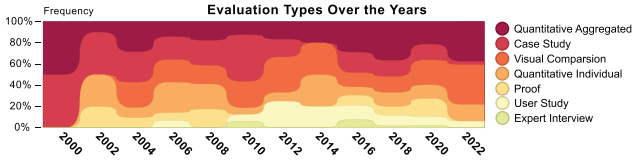
(a) A *Information availability: It is becoming more common to include supplemental material, especially in appendices. Many links from even 5 years ago are now dead.*



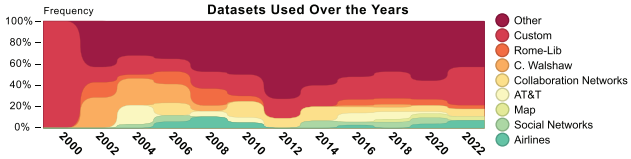
(b) A *Information availability: Around 2000, algorithms were mainly shared as pseudocode within the paper. Since 2014, it has become more common to share code in external repositories or not at all. The lack of code limits our ability to verify and reproduce published results.*



(c) M *Results measured: New ways of quantifying performance have been introduced over time, leading to a wide variety in use. Running time, while the main measure for many years, is less frequently used in recent papers.*



(d) ET *Evaluation type: There has been little change in what is used.*



(e) D *Datasets: The C. Walshaw dataset was especially popular 2002–2008, but was supplanted by other collections. New datasets and improved availability mean researchers now have a plethora of options, but some, such as Rome-Lib, have maintained their popularity over time.*

Figure 18: *How the frequency of several surveyed features have changed over time. To reduce the disparity for years in which there were fewer papers, the values shown are binned in 2-year intervals. The charts show the nine categories with the most occurrences each: everything else is grouped into 'Other'.*

The space for Results Measured expanded with a push to better understand graph aesthetics and conduct user studies. Running time is still the most reported metric, as it applies almost to every layout method. The significant change is the inclusion of other metrics to capture different aspects of a graph.

ET Evaluation types over the years have stayed mostly steady with an increase in Quantitative Aggregated Experiments in recent times; see Figure 18d. There are several possible explanations as to why this is occurring. First, the field is maturing. Similar to how more varied results are reported, it becomes easier to represent ambiguous elements like how good the layout is aesthetically through metrics. Second, accessing large data is easier than ever, and computers can run more experiments. As a culmination of both fields maturing and access to more resources, we are seeing more papers using machine learning to train layout algorithms.

14. Related Surveys

Graphs are among the most-surveyed topics, perhaps due to their wide application and complexity. We found relevant surveys through three approaches. We first examined the 86 surveys in visualization venues that were detailed in the 'Survey of Surveys' paper [ML17]. Next, we searched for relevant surveys using an online repository called 'Computer Graphics Forum: All STARs, Surveys, and Reviews' [For23]. The repository covers 199 papers from 1985 to 2020. Finally, we used Google Scholar to find surveys on graphs and networks published since 2020.

The structure, classification and ideas in this survey are based on the most relevant ones we found through our search. One notable contribution by von Landesberger *et al.* [vLKS*11] is their discussion on techniques for visual analysis of large graphs. They classify graphs according to both time dependency and structure. Time dependency can be split into static or time-dependent representations. The structure comprises trees, generic graphs (directed, undirected or mixed) and compound graphs, where multiple nodes are collapsed into a metanode representation. To provide a complete picture, they also discuss algorithms for pre-processing graphs, what interactions can be applied to graphs, and future challenges.

In a more focused investigation, Beck *et al.* [BBDW14] delve deeper into dynamic graph visualization by hierarchically categorizing publications based on their treatment of time. Either time is shown through animation, or as a static representation with a timeline. These categories are further broken down by the purpose of the layout (general or special-purpose) for animation and the type of representation for static graphs (node-link or matrix). A similarity between our work and theirs is the discussion of the evaluation process. They focus on evaluating animated approaches for dynamic graphs while we take a broader approach and examine the evaluation of all layouts.

Vehlow *et al.* [VBW17] survey visualizations that display the group features of graphs. Their taxonomy breaks down visualization techniques along two axes. One shows four main visualization categories—visual node attributes, juxtaposed, superimposed and embedded—while the other separates group structure into disjoint flat, overlapping flat, disjoint hierarchy and overlapping hierarchy. Vehlow *et al.*'s work also contains a section on evaluation through the lens of group-related tasks. They group similar tasks and summarize the results from user studies, reporting which area in group structures would benefit from further exploration.

Yoghourdjian *et al.* [YAD*18] surveyed user studies involving node-link diagrams. They wanted to better understand relative terms like 'large' or 'complex' in the context of human-centred experiments. Specifically, how different features and characteristics of a graph affect visual complexity in practice. They also outline the types of tasks and networks and their experiment design. Both of our works address how network diagrams are evaluated; while theirs focuses more on user studies, ours is evaluation in general.

Filipov *et al.* [FAM23] provide a comprehensive overview of network visualization research. Their survey consolidates various surveys and task taxonomies in this field, identifying both saturated and less-explored areas. They aim to unify terminology and categorizations across different studies, offering a structured meta-survey highlighting current research trends, commonalities and differences in network visualization.

Perhaps the survey that is closest in purpose to our own is Burch *et al.*'s [BHW*21] STAR paper on empirical user evaluation for graph visualization. They classify user evaluation based on graph interpretation, memorability and expression. Where interpretation is the ability of the user to understand the drawn graph, memorability is the ability to recall information from the graph and expression is the ability to generate the desired graph. They explicitly exclude computational experiments from their survey, which is the focus of this systematic review.

This systematic review aims to fill a gap left by these prior studies by closely examining the *computational* approaches used to evaluate graph layout algorithms. Our review complements Burch *et al.*'s [BHW*21] survey of *human-subjects* approaches to evaluation. Our report focuses on how to evaluate graph layout methods in general, while most of these existing surveys either target a specific type of graph or evaluation type. We sought answers to questions such as, 'What is the data set size used for a given graph feature', 'How reproducible is the experiment, including code and supplementary material availability', or 'What evaluation types were used and the results measured'.

15. Conclusion

Evaluating graph layout algorithms requires careful consideration of multiple aspects of the evaluation, which are all interconnected. The motivation for this study came from the authors' own experiences in running evaluations for graph layout algorithms and included discussions and proposed solutions for many of the challenges faced in their own work. The objective is to create a navigable resource for other authors that can be used to search for answers to the many questions that can come up during the development and evaluation of a layout algorithm.

As quantitative readability criteria have been defined to evaluate the quality and readability of GDs, running user studies has not

been deemed a requirement for evaluating graph layouts for quite a while. The quality and usefulness of a layout algorithm can be measured with just comparisons of numerical values, for instance, in terms of the number of crossings produced or in terms of running time. However, if we want to use computational evaluations to prove the usefulness of graph layout algorithms, we might consider moving towards the more rigid standards of fields that have used similar evaluations for a much longer time, such as those that are used in computer graphics, or for security tools. This would require a standardization process and the definition of rigid constraints. This paper does not intend to define a standard but is meant as an encouragement to make computational experiments as reproducible as possible, to allow for fair comparisons.

There is still much **future work** to be done in this field. One important aspect is the classification of datasets, which would help researchers find suitable datasets for their algorithms. It would be beneficial to provide more insights into these datasets, such as the distribution of node degrees and information about clusters within graphs. Although there are websites like Konect [Kun13] and SNAP [LS16] that offer statistics about downloadable datasets, there is currently no dedicated website specifically focused on graph layout algorithms. Additionally, most existing websites provide information about individual graphs rather than entire collections, which makes it difficult to evaluate algorithms quantitatively. Creating a website that aggregates and preserves important datasets for graph layout algorithms would be helpful.

Another crucial step is establishing clear benchmarking standards for different evaluation types. This would involve determining the minimum number of graphs and their variety needed for a meaningful benchmark. We would also need guidelines on reporting the hardware and computational resources used in evaluations. While this paper can provide resources and examples for researchers evaluating their own algorithms, it does not aim to define specific benchmarking guidelines. However, it serves as a starting point for discussions and provides valuable resources for researchers.

By addressing these areas of future work, researchers can make better decisions when selecting datasets for algorithms and ensure consistency and comparability in evaluations. This will contribute to advancements in graph layout algorithms and improve their efficiency and effectiveness. Overall, further exploration in these areas will drive the growth and development of the field.

Acknowledgements

This work was supported in part by the U.S. National Science Foundation (NSF) under award number CAREER IIS-2145382, by Northeastern University, and by Universität Konstanz. We thank Jane Kokernak for editing.

References

[ACM] ACM: ACM replicability badges (2020). https://www.acm.org/publications/policies/artifact-review-and-badging-current. [Accessed 3 Jan. 2024].

[ADLM19] Arleo A., Didimo W., Liotta G., Montecchiani F.: A distributed multilevel force-directed algorithm. *IEEE Transactions on Parallel and Distributed Systems 30*, 4 (Apr. 2019), 754–765. https://doi.org/10.1109/tpds.2018.2869805.

[ALD*20] Ahmed R., Luca F. D., Devkota S., Kobourov S., Li M.: Graph drawing via gradient descent, $(GD)^2$. In *Lecture Notes in Computer Science* (2020), Springer International Publishing, pp. 3–17. https://doi.org/10.1007/978-3-030-68766-3_1.

[BA99] Barabási A.-L., Albert R.: Emergence of scaling in random networks. *Science 286*, 5439 (1999), 509–512. https://doi.org/10.1126/science.286.5439.509.

[BBDW14] Beck F., Burch M., Diehl S., Weiskopf D.: The state of the art in visualizing dynamic graphs. In EuroVis - STARs (2014), R. Borgo, R. Maciejewski and I. Viola (Eds.), The Eurographics Association. https://doi.org/10.2312/eurovisstar.20141174.

[BCE*08] Buchheim C., Chimani M., Ebner D., Gutwenger C., Jünger M., Klau G. W., Mutzel P., Weiskircher R.: A branch-and-cut approach to the crossing number problem. *Discrete Optimization 5*, 2 (May 2008), 373–388. https://doi.org/10.1016/j.disopt.2007.05.006.

[BEJ*06] Buchheim C., Ebner D., Jünger M., Klau G. W., Mutzel P., Weiskircher R.: Exact crossing minimization. In *Graph Drawing* (Berlin, Heidelberg, 2006), P. Healyand and N. S. Nikolov (Eds.), Springer Berlin Heidelberg, pp. 37–48. https://doi.org/10.1007/11618058_4.

[Ber00] Bertault F.: A force-directed algorithm that preserves edge-crossing properties. *Information Processing Letters 74*, 1-2 (Apr. 2000), 7–13. https://doi.org/10.1016/s0020-0190(00)00042-9.

[BFG*18] Bekos M. A., Förster H., Geckeler C., Holländer L., Kaufmann M., Spallek A. M., Splett J.: A heuristic approach towards drawings of graphs with high crossing resolution. In *Lecture Notes in Computer Science* (2018). Springer International Publishing, pp. 271–285. https://doi.org/10.1007/978-3-030-04414-5_19.

[BGL*97] Battista G. D., Garg A., Liotta G., Tamassia R., Tassinari E., Vargiu F.: An experimental comparison of four graph drawing algorithms. *Computational Geometry 7*, 5-6 (Apr. 1997), 303–325. https://doi.org/10.1016/s0925-7721(96)00005-3.

[BGL*00] Battista G. D., Garg A., Liotta G., Parise A., Tamassia R., Tassinari E., Vargiu F., Vismara L.: Drawing directed acyclic graphs: An experimental study. *International Journal of Computational Geometry & Applications 10*, 06 (Dec. 2000), 623–648. https://doi.org/10.1142/s0218195900000358.

[BHW*21] Burch M., Huang W., Wakefield M., Purchase H. C., Weiskopf D., Hua J.: The state of the art in empirical user evaluation of graph visualizations. *IEEE Access 9* (2021), 4173–4198. https://doi.org/10.1109/ACCESS.2020.3047616.

[BJM02] Barth W., Jünger M., Mutzel P.: Simple and efficient bilayer cross counting. In *Graph Drawing*. M. T. Goodrich and S. G. Kobourov (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg (2002), pp. 130–141. https://doi.org/10.1007/3-540-36151-0_13.

[BK02] Brandes U., Köpf B.: Fast and simple horizontal coordinate assignment. In *Graph Drawing* (Berlin, Heidelberg, 2002), P. Mutzel, M. Jünger and S. Leipert (Eds.), Springer Berlin Heidelberg, pp. 31–44.

[BLNN21] Bhore S., Löffler M., Nickel S., Nöllenburg M.: Unit disk representations of embedded trees, outerplanar and multi-legged graphs. In *Lecture Notes in Computer Science* (2021). Springer International Publishing, pp. 304–317. https://doi.org/10.1007/978-3-030-92931-2_22.

[BM06] Batagelj V., Mrvar A.: Pajek datasets. http://vlado.fmf.uni-lj.si/pub/networks/data/ (2006). [Accessed 20 Jan. 2023].

[BP06] Brandes U., Pich C.: Eigensolver methods for progressive multidimensional scaling of large data. In GD'06: Proceedings of the 14th International Conference on Graph Drawing (2006), Springer-Verlag, Berlin, Heidelberg, pp. 42–53. https://doi.org/10.5555/1758612.1758620.

[BPF14] Bach B., Pietriga E., Fekete J.-D.: GraphDiaries: Animated transitions and temporal navigation for dynamic networks. *IEEE Transactions on Visualization and Computer Graphics 20*, 5 (May 2014), 740–754. https://doi.org/10.1109/tvcg.2013.254.

[BPWv18] Ballweg K., Pohl M., Wallner G., von Landesberger T.: Visual similarity perception of directed acyclic graphs: A study on influencing factors and similarity judgment strategies. *Journal of Graph Algorithms and Applications 22*, 3 (2018), 519–553. https://doi.org/10.7155/jgaa.00467.

[BST00] Brandes U., Shubina G., Tamassia R.: Improving angular resolution in visualizations of geographic networks. In *Data Visualization 2000* (2000), W. C.de Leeuw and R. van Liere (Eds.), Springer Vienna, Vienna, pp. 23–32. https://doi.org/10.1007/978-3-7091-6783-0_3.

[BVB*11] Burch M., Vehlow C., Beck F., Diehl S., Weiskopf D.: Parallel edge splatting for scalable dynamic graph visualization. *IEEE Transactions on Visualization and Computer Graphics 17*, 12 (Dec. 2011), 2344–2353. https://doi.org/10.1109/tvcg.2011.226.

[BW98] Brandes U., Wagner D.: Using graph layout to visualize train interconnection data. In *Graph Drawing* (Berlin, Heidelberg, 1998), S. H. Whitesides (Ed.), Springer Berlin Heidelberg, pp. 44–56. https://doi.org/10.1007/3-540-37623-2_4.

[CDMP18] Calamoneri T., Donato V. D., Mariottini D., Patrignani M.: Visualizing co-phylogenetic reconciliations. In *Lecture Notes in Computer Science* (2018). Springer International Publishing, pp. 334–347. https://doi.org/10.1007/978-3-319-73915-1_27.

[CGM06] Chimani M., Gutwenger C., Mutzel P.: Experiments on exact crossing minimization using column generation. In *Experimental Algorithms* (Berlin, Heidelberg, 2006), C. Àlvarez and M. Serna (Eds.), Springer Berlin Heidelberg, pp. 303–315. https://doi.org/10.1007/11764298_28.

[CGMW10] Chimani M., Gutwenger C., Mutzel P., Wong H.-M.: Layer-free upward crossing minimization. *ACM Journal of Experimental Algorithmics 15* (Mar. 2010). https://doi.org/10.1145/1671970.1671975.

[CGSQ11] Cao N., Gotz D., Sun J., Qu H.: DICON: Interactive visual analysis of multidimensional clusters. *IEEE Transactions on Visualization and Computer Graphics 17*, 12 (Dec. 2011), 2581–2590. https://doi.org/10.1109/tvcg.2011.188.

[CHN19] Clancy K., Haythorpe M., Newcombe A.: An effective crossing minimisation heuristic based on star insertion. *Journal of Graph Algorithms and Applications 23*, 2 (2019), 135–166. https://doi.org/10.7155/jgaa.00487

[CIW21a] Chimani M., Ilsen M., Wiedera T.: Crossing number. https://tcs.uos.de/doku.php?id=research/cr (2021). [Accessed 20 Jan. 2023].

[CIW21b] Chimani M., Ilsen M., Wiedera T.: Star-struck by fixed embeddings: Modern crossing number heuristics. In *Lecture Notes in Computer Science* (2021). Springer International Publishing, pp. 41–56. https://doi.org/10.1007/978-3-030-92931-2_3.

[CKS*16] Cornel D., Konev A., Sadransky B., Horváth Z., Brambilla A., Viola I., Waser J.: Composite flow maps. *Computer Graphics Forum 35*, 3 (2016), 461–470. https://doi.org/10.1111/cgf.12922.

[CKW16] Chimani M., Klein K., Wiedera T.: A note on the practicality of maximal planar subgraph algorithms. In *Graph Drawing and Network Visualization* (Cham, 2016), Y. Hu and M. Nöllenburg (Eds.), Springer International Publishing, pp. 357–364.

[CMB08] Chimani M., Mutzel P., Bomze I.: A new approach to exact crossing minimization. In *Algorithms - ESA 2008* (2008). Springer Berlin Heidelberg, pp. 284–296. https://doi.org/10.1007/978-3-540-87744-8_24.

[Coh97] Cohen J. D.: Drawing graphs to convey proximity: An incremental arrangement method. *ACM Transactions on Computer-Human Interaction 4*, 3 (Sep. 1997), 197–229. https://doi.org/10.1145/264645.264657.

[Coh22] Cohen W. W.: Enron email dataset. https://www.cs.cmu.edu/~enron/ (2022). [Accessed 20 Jan. 2023].

[CPPS19] Chen F., Piccinini L., Poncelet P., Sallaberry A.: Node overlap removal algorithms: A comparative study. In *Lecture Notes in Computer Science* (2019). Springer International Publishing, pp. 179–192. https://doi.org/10.1007/978-3-030-35802-0_14.

[CvGM*18] Castermans T., van Garderen M., Meulemans W., Nöllenburg M., Yuan X.: Short plane supports for spatial hypergraphs. In *Lecture Notes in Computer Science* (2018). Springer International Publishing, pp. 53–66. https://doi.org/10.1007/978-3-030-04414-5_4.

[CW16] Chimani M., Wiedera T.: An ILP-based proof system for the crossing number problem. In 24th Annual European Symposium on Algorithms (ESA 2016) (Dagstuhl, Germany, 2016), vol. *57* of *Leibniz International Proceedings in Informatics (LIPIcs)*, P. Sankowski and C. Zaroliagis (Eds.), Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, pp. 29:1–29:13. https://doi.org/10.4230/LIPIcs.ESA.2016.29.

[CZQ*08] Cui W., Zhou H., Qu H., Wong P. C., Li X.: Geometry-based edge clustering for graph visualization. *IEEE Transactions on Visualization and Computer Graphics 14*, 6 (Nov. 2008), 1277–1284. https://doi.org/10.1109/tvcg.2008.135.

[DAL*19] Devkota S., Ahmed R., Luca F. D., Isaacs K. E., Kobourov S.: Stress-Plus-X (SPX) graph layout. In *Lecture Notes in Computer Science* (2019). Springer International Publishing, pp. 291–304. https://doi.org/10.1007/978-3-030-35802-0_23.

[DBCSD23] Di Bartolomeo S., Crnovrsanin T., Saffo D., Dunne C.: Supplemental material location. https://osf.io/dn5zq/ (2023). [Accessed 3 Jan. 2024].

[DBPB*22] Di Bartolomeo S., Pister A., Buono P., Plaisant C., Dunne C., Fekete J.-D.: Six methods for transforming layered hypergraphs to apply layered graph layout algorithms. *Computer Graphics Forum 41*, 3 (2022), 259–270. https://doi.org/10.1111/cgf.14538.

[DBPW*23] Di Bartolomeo S., Puerta E., Wilson C., Crnovrsanin T., Dunne C.: A collection of benchmark datasets for evaluating graph layout algorithms. https://doi.org/10.31219/osf.io/yftju (Sep. 2023).

[DBRGD22] Di Bartolomeo S., Riedewald M., Gatterbauer W., Dunne C.: STRATISFIMAL LAYOUT: A modular optimization model for laying out layered node-link network visualizations. *IEEE Transactions on Visualization and Computer Graphics 28*, 1 (Jan. 2022), 324–334. https://doi.org/10.1109/tvcg.2021.3114756.

[dBZSD21] di Bartolomeo S., Zhang Y., Sheng F., Dunne C.: Sequence braiding: Visual overviews of temporal event sequences and attributes. *IEEE Transactions on Visualization and Computer Graphics 27*, 2 (Feb. 2021), 1353–1363. https://doi.org/10.1109/tvcg.2020.3030442.

[dCKN19] de Col P., Klute F., Nöllenburg M.: Mixed linear layouts: Complexity, heuristics, and experiments. In *Lecture Notes in Computer Science* (2019). Springer International Publishing, pp. 460–467. https://doi.org/10.1007/978-3-030-35802-0_35.

[DGNS13] Duncan C. A., Gutwenger C., Nachmanson L., Sander G.: Graph drawing contest report. In *Graph Drawing* (Berlin, Heidelberg, 2013), W. Didimo and M. Patrignani (Eds.), Springer Berlin Heidelberg, pp. 575–579. https://doi.org/10.1007/978-3-642-36763-2_58.

[DH11] Davis T. A., Hu Y.: The university of Florida sparse matrix collection. *ACM Transactions on Mathematical Software 38*, 1 (Dec. 2011). https://doi.org/10.1145/2049662.2049663.

[DI18] Devkota S., Isaacs K. E.: CFGExplorer: Designing a visual control flow analytics system around basic program analysis operations. *Computer Graphics Forum 37*, 3 (2018), 453–464. https://doi.org/10.1111/cgf.13433.

[DL13] Didimo W., Liotta G.: *The Crossing-Angle Resolution in Graph Drawing*. Springer New York, New York, NY, 2013, pp. 167–184. https://doi.org/10.1007/978-1-4614-0110-0_10.

[DPF16] Dang T. N., Pendar N., Forbes A. G.: TimeArcs: Visualizing fluctuations in dynamic networks. *Computer Graphics Forum 35*, 3 (June 2016), 61–69. https://doi.org/10.1111/cgf.12882.

[Dra22] Drawing G.: Graph drawing website. http://www.graphdrawing.org/data.html (2022). [Accessed 20 Jan. 2023].

[DRMM13] Dwyer T., Riche N. H., Marriott K., Mears C.: Edge compression techniques for visualization of dense directed graphs. *IEEE Transactions on Visualization and Computer Graphics 19*, 12 (Dec. 2013), 2596–2605. https://doi.org/10.1109/tvcg.2013.151.

[DRSM15] Dunne C., Ross S. I., Shneiderman B., Martino M.: Readability metric feedback for aiding node-link visualization designers. *IBM Journal of Research and Development59*, *2/3* (2015), 14:1–14:16. https://doi.org/10.1147/JRD.2015.2411412.

[EASG*17] El-Assady M., Sevastjanova R., Gipp B., Keim D., Collins C.: NEREx: Named-entity relationship exploration in multi-party conversations. *Computer Graphics Forum 36*, 3 (2017), 213–225. https://doi.org/10.1111/cgf.13181.

[EHP*11] Ersoy O., Hurter C., Paulovich F. V., Cantareiro G., Telea A.: Skeleton-based edge bundling for graph visualization. *IEEE Transactions on Visualization and Computer Graphics 17*, 12 (Dec. 2011), 2364–2373. https://doi.org/10.1109/tvcg.2011.233.

[ENH18] Eades P., Nguyen Q., Hong S.-H.: Drawing big graphs using spectral sparsification. In *Lecture Notes in Computer Science* (2018). Springer International Publishing, pp. 272–286. https://doi.org/10.1007/978-3-319-73915-1_22.

[ER*60] Erdős P., Rényi A.: On the evolution of random graphs. *Publications of the Mathematical Institute of the Hungarian Academy of Sciences 5*, 1 (1960), 17–60.

[FAM23] Filipov V., Arleo A., Miksch S.: Are we there yet? A roadmap of network visualization from surveys to task

taxonomies. *Computer Graphics Forum 42*, 6 (2023), e14794. https://doi.org/10.1111/cgf.14794.

[FHH*93] Formann M., Hagerup T., Haralambides J., Kaufmann M., Leighton F. T., Symvonis A., Welzl E., Woeginger G.: Drawing graphs in the plane with high resolution. *SIAM Journal on Computing 22*, 5 (1993), 1035–1052. https://doi.org/10.1137/0222063.

[FKR21] Förster H., Kaufmann M., Raftopoulou C. N.: Recognizing and embedding simple optimal 2-planar graphs. In *Lecture Notes in Computer Science* (2021). Springer International Publishing, pp. 87–100. https://doi.org/10.1007/978-3-030-92931-2_6.

[FLM95] Frick A., Ludwig A., Mehldau H.: A fast adaptive layout algorithm for undirected graphs (extended abstract and system demonstration). In *Graph Drawing* (Berlin, Heidelberg, 1995), R. Tamassia and I. G. Tollis (Eds.), Springer Berlin Heidelberg, pp. 388–403. https://doi.org/10.1007/3-540-58950-3_393.

[For05] Forster M.: A fast and simple heuristic for constrained two-level crossing reduction. In *Graph Drawing* (2005). Springer Berlin Heidelberg, pp. 206–216. https://doi.org/10.1007/978-3-540-31843-9_22.

[For23] Forum C. G.: Computer graphics forum: All stars, surveys, and reviews. https://sites.google.com/site/drminchen/cgf-info/cgf-stars (2023). [Accessed Jan. 2023].

[FR91a] Fruchterman T. M. J., Reingold E. M.: Graph drawing by force-directed placement. *Software: Practice and Experience 21*, 11 (1991), 1129–1164. https://doi.org/10.1002/spe.4380211102.

[FT04] Frishman Y., Tal A.: Dynamic drawing of clustered graphs. In IEEE Symposium on Information Visualization (2004), pp. 191–198. https://doi.org/10.1109/INFVIS.2004.18.

[FT07] Frishman Y., Tal A.: Multi-level graph layout on the GPU. *IEEE Transactions on Visualization and Computer Graphics 13*, 6 (2007), 1310–1319. https://doi.org/10.1109/TVCG.2007.70580.

[FT08] Frishman Y., Tal A.: Online dynamic graph drawing. *IEEE Transactions on Visualization and Computer Graphics 14*, 4 (July 2008), 727–740. https://doi.org/10.1109/tvcg.2008.11.

[GAM14] Grabowicz P. A., Aiello L. M., Menczer F.: Fast filtering and animation of large dynamic networks. *EPJ Data Science 3*, 1 (Oct. 2014). https://doi.org/10.1140/epjds/s13688-014-0027-8.

[GBD09] Greilich M., Burch M., Diehl S.: Visualizing the evolution of compound digraphs with timeArcTrees. *Computer Graphics Forum 28*, 3 (2009), 975–982. https://doi.org/10.1111/j.1467-8659.2009.01451.x.

[GDL*20] Giacomo E. D., Didimo W., Liotta G., Montecchiani F., Tappini A.: Storyline visualizations with ubiquitous actors. In *Lecture Notes in Computer Science* (2020). Springer International Publishing, pp. 324–332. https://doi.org/10.1007/978-3-030-68766-3_25.

[GHN13] Gansner E. R., Hu Y., North S.: A maxent-stress model for graph layout. *IEEE Transactions on Visualization and Computer Graphics 19*, 6 (June 2013), 927–940. https://doi.org/10.1109/tvcg.2012.299.

[GJLM16] Gronemann M., Jünger M., Liers F., Mambelli F.: Crossing minimization in storyline visualization. In *Lecture Notes in Computer Science* (2016). Springer International Publishing, pp. 367–381. https://doi.org/10.1007/978-3-319-50106-2_29.

[GK00] Gajer P., Kobourov S. G.: Grip: Graph drawing with intelligent placement. In GD'00: Proceedings of the 8th International Symposium on Graph Drawing (Berlin, Heidelberg, 2000), Springer-Verlag, pp. 222–228. https://doi.org/10.5555/647552.729406.

[GKN05a] Gansner E., Koren Y., North S.: Topological fisheye views for visualizing large graphs. *IEEE Transactions on Visualization and Computer Graphics 11*, 4 (July 2005), 457–468. https://doi.org/10.1109/tvcg.2005.66.

[GKN05b] Gansner E. R., Koren Y., North S.: Graph drawing by stress majorization. In *Graph Drawing* (2005). Springer Berlin Heidelberg, pp. 239–250. https://doi.org/10.1007/978-3-540-31843-9_25.

[GLA*21] Giovannangeli L., Lalanne F., Auber D., Giot R., Bourqui R.: Deep neural network for DrawiNg networks. In *Lecture Notes in Computer Science* (2021). Springer International Publishing, pp. 375–390. https://doi.org/10.1007/978-3-030-92931-2_27.

[GLNR14] Gutwenger C., Löffler M., Nachmanson L., Rutter I.: Graph drawing contest report. In *Graph Drawing* (Berlin, Heidelberg, 2014), C. Duncan and A. Symvonis (Eds.), Springer Berlin Heidelberg, pp. 501–506. https://doi.org/10.1007/978-3-662-45803-7_42.

[GN02] Girvan M., Newman M. E. J.: Community structure in social and biological networks. *Proceedings of the National Academy of Sciences 99*, 12 (2002), 7821–7826. https://doi.org/10.1073/pnas.122653799.

[Gra08] Graphviz: Graphviz gallery. https://graphviz.org/gallery/ (2008). [Accessed 20 Jan. 2023].

[GSM11] Gange G., Stuckey P. J., Marriott K.: Optimal k-level planarization and crossing minimization. In *Graph Drawing* (2011). Springer Berlin Heidelberg, pp. 238–249. https://doi.org/10.1007/978-3-642-18469-7_22.

[Har18] Haroz S.: Open practices in visualization research: Opinion paper. In 2018 IEEE Evaluation and Beyond — Methodological Approaches for Visualization (BELIV) (2018), pp. 46–52. https://doi.org/10.1109/BELIV.2018.8634427.

[HBH18] Hoffswell J., Borning A., Heer J.: Setcola: High-level constraints for graph layout. *Computer Graphics Forum 37*, 3 (2018), 537–548. https://doi.org/10.1111/cgf.13440.

[HET*19] Hong S.-H., Eades P., Torkel M., Wang Z., Chae D., Hong S., Langerenken D., Chafi H.: Multi-level graph drawing using infomap clustering. In *Lecture Notes in Computer Science* (2019). Springer International Publishing, pp. 139–146. https://doi.org/10.1007/978-3-030-35802-0_11.

[HFM07] Henry N., Fekete J.-D., McGuffin M. J.: NodeTrix: a hybrid visualization of social networks. *IEEE Transactions on Visualization and Computer Graphics 13*, 6 (Nov. 2007), 1302–1309. https://doi.org/10.1109/tvcg.2007.70582.

[HHE08] Huang W., Hong S.-H., Eades P.: Effects of crossing angles. In 2008 IEEE Pacific Visualization Symposium (2008), pp. 41–46. https://doi.org/10.1109/PACIFICVIS.2008.4475457.

[HJ05a] Hachul S., Jünger M.: Drawing large graphs with a potential-field-based multilevel algorithm. In *Graph Drawing* (Berlin, Heidelberg, 2005), J. Pach (Ed.), Springer Berlin Heidelberg, pp. 285–295. https://doi.org/10.1007/978-3-540-31843-9_29.

[HJ05b] Hachul S., Jünger M.: Large-Graph Layout with the Fast Multipole Multilevel Method. Working paper, Universität zu Köln, 2005. https://kups.ub.uni-koeln.de/54892/.

[HJ06] Hachul S., Jünger M.: An experimental comparison of fast algorithms for drawing general large graphs. In *Graph Drawing* (2006). Springer Berlin Heidelberg, pp. 235–250. https://doi.org/10.1007/11618058_22.

[HKNN19] Hummel M., Klute F., Nickel S., Nöllenburg M.: Maximizing ink in partial edge drawings of k-plane graphs. In *Lecture Notes in Computer Science* (2019). Springer International Publishing, pp. 323–336. https://doi.org/10.1007/978-3-030-35802-0_25.

[Hu05] Hu Y. F.: Efficient and high quality force-directed graph drawing. *The Mathematica Journal 10* (2005), 37–71. http://www.mathematica-journal.com/issue/v10i1/contents/graph_draw/graph_draw.pdf.

[HvW09] Holten D., van Wijk J. J.: Force-directed edge bundling for graph visualization. *Computer Graphics Forum 28*, 3 (June 2009), 983–990. https://doi.org/10.1111/j.1467-8659.2009.01450.x.

[HZM*16] Huang X., Zhao Y., Ma C., Yang J., Ye X., Zhang C.: TrajGraph: A graph-based visual analytics approach to studying urban network centralities using taxi trajectory data. *IEEE Transactions on Visualization and Computer Graphics 22*, 1 (Jan. 2016), 160–169. https://doi.org/10.1109/tvcg.2015.2467771.

[IHK*17] Isenberg P., Heimerl F., Koch S., Isenberg T., Xu P., Stolper C., Sedlmair M., Chen J., Möller T., Stasko J.: Vispubdata.org: A metadata collection about IEEE visualization (VIS) publications. *IEEE Transactions on Visualization and Computer Graphics 23*, 9 (Sept. 2017), 2199–2206. https://doi.org/10.1109/TVCG.2016.2615308.

[JM96] Jünger M., Mutzel P.: Exact and heuristic algorithms for 2-layer straightline crossing minimization. In *Graph Drawing* (Berlin, Heidelberg, 1996), F. J. Brandenburg (Ed.), Springer Berlin Heidelberg, pp. 337–348. https://doi.org/10.1007/BFb0021817.

[JMM*16] Jabrayilov A., Mallach S., Mutzel P., Rüegg U., von Hanxleden R.: Compact layered drawings of general directed graphs. In *Lecture Notes in Computer Science* (2016). Springer International Publishing, pp. 209–221. https://doi.org/10.1007/978-3-319-50106-2_17.

[JMS18] Jünger M., Mutzel P., Spisla C.: A flow formulation for horizontal coordinate assignment with prescribed width. In *Lecture Notes in Computer Science* (2018). Springer International Publishing, pp. 187–199. https://doi.org/10.1007/978-3-030-04414-5_13.

[KBI*18] Kim N. W., Bach B., Im H., Schriber S., Gross M., Pfister H.: Visualizing nonlinear narratives with story curves. *IEEE Transactions on Visualization and Computer Graphics 24*, 1 (2018), 595–604. https://doi.org/10.1109/TVCG.2017.2744118.

[KCH02] Koren Y., Carmel L., Harel D.: ACE: a fast multiscale eigenvectors computation for drawing huge graphs. In IEEE Symposium on Information Visualization, 2002. INFOVIS 2002 (2002), pp. 137–144. https://doi.org/10.1109/INFVIS.2002.1173159.

[KKRS21] Klemz B., Knorr K., Reddy M. M., Schröder F.: Simplifying non-simple fan-planar drawings. In *Lecture Notes in Computer Science* (2021). Springer International Publishing, pp. 57–71. https://doi.org/10.1007/978-3-030-92931-2_4.

[KMP18] Klammler M., Mchedlidze T., Pak A.: Aesthetic discrimination of graph layouts. In *Graph Drawing and Network Visualization* (Cham, 2018), T. Biedl and A. Kerren (Eds.), Springer International Publishing, pp. 169–184. https://doi.org/10.1007/978-3-030-04414-5_12.

[KMS18] Kindermann P., Meulemans W., Schulz A.: Experimental analysis of the accessibility of drawings with few segments. *Journal of Graph Algorithms and Applications 22*, 3 (2018), 501–518. https://doi.org/10.7155/jgaa.00474.

[Kor05] Koren Y.: Drawing graphs by eigenvectors: Theory and practice. *Computers & Mathematics with Applications 49*, 11 (2005), 1867–1888. https://doi.org/10.1016/j.camwa.2004.08.015.

[KPK18] Kucher K., Paradis C., Kerren A.: The state of the art in sentiment visualization. *Computer Graphics Forum 37*, 1 (2018), 71–96. https://doi.org/10.1111/cgf.13217.

[KRM*17] Kruiger J. F., Rauber P. E., Martins R. M., Kerren A., Kobourov S., Telea A. C.: Graph layouts by t-SNE.

Computer Graphics Forum 36, 3 (2017), 283–294. https://doi.org/10.1111/cgf.13187.

[Kun13] Kunegis J.: KONECT—The Koblenz Network Collection. In Proceedings of the International Conference on World Wide Web Companion (2013), pp. 1343–1350. http://dl.acm.org/citation.cfm?id=2488173.

[LEN05] Li W., Eades P., Nikolov N.: Using spring algorithms to remove node overlapping. In APVis'05: Proceedings of the 2005 Asia-Pacific Symposium on Information Visualisation—Volume 45 (AUS, 2005), Australian Computer Society, Inc., pp. 131–140. https://doi.org/10.5555/1082315.1082334.

[LHT17] Lhuillier A., Hurter C., Telea A.: State of the art in edge and trail bundling techniques. *Computer Graphics Forum 36*, 3 (2017), 619–645. https://doi.org/10.1111/cgf.13213.

[LS16] Leskovec J., Sosič R.: Snap: A general-purpose network analysis and graph-mining library. *ACM Transactions on Intelligent Systems and Technology (TIST) 8*, 1 (2016), 1–20. https://doi.org/10.1145/2898361.

[LSCL10] Lin Y.-R., Sun J., Cao N., Liu S.: ContexTour: Contextual contour visual analysis on dynamic multi-relational clustering. In Proceedings of the 2010 SIAM International Conference on Data Mining (Apr. 2010), Society for Industrial and Applied Mathematics. https://doi.org/10.1137/1.9781611972801.37.

[LSL*17] Liu M., Shi J., Li Z., Li C., Zhu J., Liu S.: Towards better analysis of deep convolutional neural networks. *IEEE Transactions on Visualization and Computer Graphics 23*, 1 (Jan. 2017), 91–100. https://doi.org/10.1109/tvcg.2016.2598831.

[LWW*13] Liu S., Wu Y., Wei E., Liu M., Liu Y.: StoryFlow: Tracking the evolution of stories. *IEEE Transactions on Visualization and Computer Graphics 19*, 12 (Dec. 2013), 2436–2445. https://doi.org/10.1109/tvcg.2013.196.

[MA19] Misue K., Akasaka K.: Graph drawing with morphing partial edges. In *Lecture Notes in Computer Science*. Springer International Publishing (2019), pp. 337–349. https://doi.org/10.1007/978-3-030-35802-0_26.

[Mal19] Mallach S.: A natural quadratic approach to the generalized graph layering problem. In *Lecture Notes in Computer Science* (2019). Springer International Publishing, pp. 532–544. https://doi.org/10.1007/978-3-030-35802-0_40.

[MHE20] Meidiana A., Hong S.-H., Eades P.: New quality metrics for dynamic graph drawing. In *Graph Drawing and Network Visualization* (Cham, 2020), D. Auber and P. Valtr (Eds.), Springer International Publishing, pp. 450–465. https://doi.org/10.1007/978-3-030-68766-3_35.

[MHEK19] Meidiana A., Hong S.-H., Eades P., Keim D.: A quality metric for visualization of clusters in graphs. In *Lecture Notes in Computer Science* (2019). Springer International Publishing, pp. 125–138. https://doi.org/10.1007/978-3-030-35802-0_10.

[ML17] McNabb L., Laramee R. S.: Survey of surveys (SoS)—mapping the landscape of survey papers in information visualization. *Computer Graphics Forum 36*, 3 (2017), 589–617. https://doi.org/10.1111/cgf.13212.

[MM08a] Muelder C., Ma K.-L.: Rapid graph layout using space filling curves. *IEEE Transactions on Visualization and Computer Graphics 14*, 6 (2008), 1301–1308. https://doi.org/10.1109/TVCG.2008.158.

[MM08b] Muelder C., Ma K.-L.: A treemap based method for rapid layout of large graphs. In 2008 IEEE Pacific Visualization Symposium (2008), pp. 231–238. https://doi.org/10.1109/PACIFICVIS.2008.4475481.

[MRS*13] Meulemans W., Riche N. H., Speckmann B., Alper B., Dwyer T.: KelpFusion: A hybrid set visualization technique. *IEEE Transactions on Visualization and Computer Graphics 19*, 11 (Nov. 2013), 1846–1858. https://doi.org/10.1109/tvcg.2013.76.

[MST*14] Marner M. R., Smith R. T., Thomas B. H., Klein K., Eades P., Hong S.-H.: GION: Interactively untangling large graphs on wall-sized displays. In *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications*. Springer International Publishing, Cham (2014), pp. 113–124. https://doi.org/10.1007/978-3-662-45803-7_10.

[Mä90] Mäkinen E.: How to draw a hypergraph. *International Journal of Computer Mathematics 34*, 3-4 (1990), 177–185. https://doi.org/10.1080/00207169008803875.

[NEH13] Nguyen Q., Eades P., Hong S.-H.: On the faithfulness of graph visualizations. In 2013 IEEE Pacific Visualization Symposium (PacificVis) (2013), pp. 209–216. https://doi.org/10.1109/PacificVis.2013.6596147.

[NIS07] NIST: Matrix market. https://math.nist.gov/MatrixMarket/ (2007). [Accessed 20 Jan. 2023].

[NNB*17] Nachmanson L., Nocaj A., Bereg S., Zhang L., Holroyd A.: Node overlap removal by growing a tree. *Journal of Graph Algorithms and Applications 21*, *5* (2017), 857–872. https://doi.org/10.7155/jgaa.00442.

[Noa04] Noack A.: An energy model for visual graph clustering. In *Graph Drawing* (Berlin, Heidelberg, 2004), G. Liotta (Ed.), Springer Berlin Heidelberg, pp. 425–436. https://doi.org/10.1007/978-3-540-24595-7_40.

[NOB15] Nocaj A., Ortmann M., Brandes U.: Untangling the hairballs of multi-centered, small-world online social media networks. *Journal of Graph Algorithms and Applications: JGAA 19*, 2 (2015), 595–618. https://doi.org/10.7155/jgaa.00370.

[NR20] Niedermann B., Rutter I.: An integer-linear program for bend-minimization in ortho-radial drawings. In *Lecture Notes in Computer Science* (2020). Springer International Publishing, pp. 235–249. https://doi.org/10.1007/978-3-030-68766-3_19.

[NRS08] Navlakha S., Rastogi R., Shrivastava N.: Graph summarization with bounded error. In SIGMOD'08: Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data (New York, NY, USA, 2008), Association for Computing Machinery, pp. 419–432. https://doi.org/10.1145/1376616.1376661.

[NSM*19] Nickel S., Sondag M., Meulemans W., Chimani M., Kobourov S., Peltonen J., Nöllenburg M.: Computing stable demers cartograms. In *Lecture Notes in Computer Science* (2019). Springer International Publishing, pp. 46–60. https://doi.org/10.1007/978-3-030-35802-0_4.

[NW11] Nollenburg M., Wolff A.: Drawing and labeling high-quality metro maps by mixed-integer programming. *IEEE Transactions on Visualization and Computer Graphics 17*, 5 (May 2011), 626–641. https://doi.org/10.1109/tvcg.2010.81.

[OKB17] Ortmann M., Klimenta M., Brandes U.: A sparse stress model. *Journal of Graph Algorithms and Applications 21*, 5 (2017), 791–821. https://doi.org/10.7155/jgaa.00440.

[OM10] Ogawa M., Ma K.-L.: Software evolution storylines. In Proceedings of the 5th international symposium on Software visualization—SOFTVIS '10 (2010), ACM Press. https://doi.org/10.1145/1879211.1879219.

[OT18] Ortali G., Tollis I. G.: Algorithms and bounds for drawing directed graphs. In *Lecture Notes in Computer Science* (2018). Springer International Publishing, pp. 579–592. https://doi.org/10.1007/978-3-030-04414-5_41.

[PAK*20] Purchase H. C., Archambault D., Kobourov S., Nöllenburg M., Pupyrev S., Wu H.-Y.: The turing test for graph drawing algorithms. In *Lecture Notes in Computer Science* (2020). Springer International Publishing, pp. 466–481. https://doi.org/10.1007/978-3-030-68766-3_36.

[PBH18] Padia K., Bandara K., Healey C.: Yarn: Generating storyline visualizations using HTN planning. In Proceedings of Graphics Interface 2018 Toronto (2018), pp. 26–33. https://doi.org/10.20380/GI2018.05.

[PBH19] Padia K., Bandara K. H., Healey C. G.: A system for generating storyline visualizations using hierarchical task network planning. *Computers & Graphics 78* (Feb. 2019), 64–75. https://doi.org/10.1016/j.cag.2018.11.004.

[PT00] Pach J., Tóth G.: Which crossing number is it anyway? *Journal of Combinatorial Theory Series B 80*, 2 (Nov. 2000), 225–246. https://doi.org/10.1006/jctb.2000.1978.

[Pur97] Purchase H.: Which aesthetic has the greatest effect on human understanding? In *Graph Drawing* (Berlin, Heidelberg, 1997), Springer Berlin Heidelberg, pp. 248–261. https://doi.org/10.1007/3-540-63938-1_67.

[Pur02] Purchase H. C.: Metrics for graph drawing aesthetics. *Journal of Visual Languages & Computing 13*, 5 (2002), 501–516. https://doi.org/10.1006/jvlc.2002.0232.

[QZZ22] Qu B., Zhang E., Zhang Y.: Automatic polygon layout for primal-dual visualization of hypergraphs. *IEEE Transactions on Visualization and Computer Graphics 28*, 1 (Jan. 2022), 633–642. https://doi.org/10.1109/tvcg.2021.3114759.

[RESvH16] Rüegg U., Ehlers T., Spönemann M., von Hanxleden R.: A generalization of the directed graph layering problem. In *Lecture Notes in Computer Science* (2016). Springer International Publishing, pp. 196–208. https://doi.org/10.1007/978-3-319-50106-2_16.

[RW18] Raj M., Whitaker R. T.: Anisotropic radial layout for visualizing centrality and structure in graphs. In *Lecture Notes in Computer Science* (2018). Springer International Publishing, pp. 351–364. https://doi.org/10.1007/978-3-319-73915-1_28.

[SAK18] Simonetto P., Archambault D., Kobourov S.: Drawing dynamic graphs without timeslices. In *Lecture Notes in Computer Science* (2018). Springer International Publishing, pp. 394–409. https://doi.org/10.1007/978-3-319-73915-1_31.

[SIG] SIGMOD: Sigmod availability and reproducibility initiative (2020). https://reproducibility.sigmod.org/. [Accessed 3 Jan. 2024].

[SSS*12] Strobelt H., Spicker M., Stoffel A., Keim D., Deussen O.: Rolled-out wordles: A heuristic method for overlap removal of 2D data representatives. *Computer Graphics Forum 31*, 3pt3 (June 2012), 1135–1144. https://doi.org/10.1111/j.1467-8659.2012.03106.x.

[Sta] Stamp G. R.: Graphics replicability stamp (2022). https://www.replicabilitystamp.org/. [Accessed 3 Jan. 2024].

[STT81] Sugiyama K., Tagawa S., Toda M.: Methods for visual understanding of hierarchical system structures. *IEEE Transactions on Systems, Man, and Cybernetics 11*, 2 (1981), 109–125. https://doi.org/10.1109/TSMC.1981.4308636.

[THM15] Tanahashi Y., Hsueh C.-H., Ma K.-L.: An efficient framework for generating storyline visualizations from streaming data. *IEEE Transactions on Visualization and Computer Graphics 21*, 6 (June 2015), 730–742. https://doi.org/10.1109/tvcg.2015.2392771.

[TM12] Tanahashi Y., Ma K.-L.: Design considerations for optimizing storyline visualizations. *IEEE Transactions on Visualization and Computer Graphics 18*, 12 (Dec. 2012), 2679–2688. https://doi.org/10.1109/tvcg.2012.212.

[VAA*14] Vines T., Albert A., Andrew R., Débarre F., Bock D., Franklin M., Gilbert K., Moore J.-S., Renaut S., Rennison D.: The availability of research data declines rapidly with article age. *Current Biology 24*, 1 (2014), 94–97. https://doi.org/10.1016/j.cub.2013.11.014.

[VBW17] Vehlow C., Beck F., Weiskopf D.: Visualizing group structures in graphs: A survey. *Computer Graphics Forum 36*, 6 (2017), 201–225. https://doi.org/10.1111/cgf.12872.

[vdEHBvW13] van den Elzen S., Holten D., Blaas J., van Wijk J. J.: Reordering massive sequence views: Enabling temporal and structural analysis of dynamic networks. In 2013 IEEE Pacific Visualization Symposium (PacificVis) (Feb. 2013), IEEE. https://doi.org/10.1109/pacificvis.2013.6596125.

[vDLMW18] van Dijk T. C., Lipp F., Markfelder P., Wolff A.: Computing storyline visualizations with few block crossings. In *Lecture Notes in Computer Science* (2018). Springer International Publishing, pp. 365–378. https://doi.org/10.1007/978-3-319-73915-1_29

[vLBR*16] von Landesberger T., Brodkorb F., Roskosch P., Andrienko N., Andrienko G., Kerren A.: MobilityGraphs: Visual analysis of mass mobility dynamics via spatio-temporal graphs and clustering. *IEEE Transactions on Visualization and Computer Graphics 22*, 1 (Jan. 2016), 11–20. https://doi.org/10.1109/tvcg.2015.2468111.

[vLKS*11] von Landesberger T., Kuijper A., Schreck T., Kohlhammer J., van Wikj J., Fekete J.-D., Fellner D.: Visual analysis of large graphs: State-of-the-art and future research challenges. *Computer Graphics Forum 30*, 6 (2011), 1719–1749. https://doi.org/10.1111/j.1467-8659.2011.01898.x.

[WAA*22] Wallinger M., Archambault D., Auber D., Nollenburg M., Peltonen J.: Edge-path bundling: A less ambiguous edge bundling approach. *IEEE Transactions on Visualization and Computer Graphics 28*, 1 (Jan. 2022), 313–323. https://doi.org/10.1109/tvcg.2021.3114795.

[Wal01a] Walshaw C.: C. Walshaw's graph collection. https://chriswalshaw.co.uk/partition/ (2001). [Accessed 20 Jan. 2023].

[Wal01b] Walshaw C.: A multilevel algorithm for force-directed graph drawing. In *Graph Drawing* (Berlin, Heidelberg, 2001), J. Marks (Ed.), Springer Berlin Heidelberg, pp. 171–182.

[WSW*18] Wongsuphasawat K., Smilkov D., Wexler J., Wilson J., Mane D., Fritz D., Krishnan D., Viegas F. B., Wattenberg M.: Visualizing dataflow graphs of deep learning models in TensorFlow. *IEEE Transactions on Visualization and Computer Graphics 24*, 1 (Jan. 2018), 1–12. https://doi.org/10.1109/tvcg.2017.2744878.

[WWS*18] Wang Y., Wang Y., Sun Y., Zhu L., Lu K., Fu C.-W., Sedlmair M., Deussen O., Chen B.: Revisiting stress majorization as a unified framework for interactive constrained graph visualization. *IEEE Transactions on Visualization and Computer Graphics 24*, 1 (2018), 489–499. https://doi.org/10.1109/TVCG.2017.2745919.

[WXW*20] Wang Y., Xue M., Wang Y., Yan X., Chen B., Fu C.-W., Hurter C.: Interactive structure-aware blending of diverse edge bundling visualizations. *IEEE Transactions on Visualization and Computer Graphics 26*, 1 (2020), 687–696. https://doi.org/10.1109/TVCG.2019.2934805.

[WZBW20] Walter J., Zink J., Baumeister J., Wolff A.: Layered drawing of undirected graphs with generalized port constraints. In *Lecture Notes in Computer Science* (2020). Springer International Publishing, pp. 220–234. https://doi.org/10.1007/978-3-030-68766-3_18.

[WZZY18] Wu J., Zeng J., Zhu F., Yu H.: MLSEB: Edge bundling using moving least squares approximation. In *Lecture Notes in Computer Science* (2018). Springer International Publishing, pp. 379–393. https://doi.org/10.1007/978-3-319-73915-1_30.

[YAD*18] Yoghourdjian V., Archambault D., Diehl S., Dwyer T., Klein K., Purchase H. C., Wu H.-Y.: Exploring the limits of complexity: A survey of empirical studies on graph visualisation. *Visual Informatics 2*, 4 (2018), 264–282. https://doi.org/10.1016/j.visinf.2018.12.006.

[ZJC*21] Zhao Y., Jiang H., Chen Q., Qin Y., Xie H., Wu Y., Liu S., Zhou Z., Xia J., Zhou F.: Preserving minority structures in graph sampling. *IEEE Transactions on Visualization and Computer Graphics 27*, 2 (Feb. 2021), 1698–1708. https://doi.org/10.1109/tvcg.2020.3030428.

[ZSJT19] Zeng W., Shen Q., Jiang Y., Telea A.: Route-aware edge bundling for visualizing origin-destination trails in urban traffic. *Computer Graphics Forum 38*, 3 (2019), 581–593. https://doi.org/10.1111/cgf.13712.