Universal Quality Metrics for Graph Drawings: Which Graphs Excite Us Most?

Gavin J. Mooney
Monash University, Melbourne, Australia

Tim Hegemann
Universität Würzburg, Germany

Alexander Wolff
Universität Würzburg, Germany

Michael Wybrow
Monash University, Melbourne, Australia

Helen C. Purchase
Monash University, Melbourne, Australia

Abstract

Graphs are drawn for various purposes, and drawings are meant to display various features of a graph (such as planarity, Hamiltonicity). Still, there is a long history in measuring the quality of a graph drawing. Most of the metrics that have been implemented and used in large studies assume that graphs are drawn straight-line. Most of the studies use randomly generated graphs or one of very few existing benchmark sets that consist of graphs with a specific technical background (e.g., telecommunication networks).

In this paper, we extend ten commonly used metrics to node-link diagrams where edges can be curves or polygonal chains. We implement these measures and use them to evaluate a new collection of graph drawings that we have extracted from 27 proceedings of the Graph Drawing conference using an automated pipeline. We compare the "metrics landscape" of our new benchmark set, the *GD-collection-v1*, which seems to mostly contain manually drawn graphs, to the metric landscape of a benchmark set with randomly generated graphs and computer-generated straight-line drawings that has been used in a recent study [Mooney et al.; PacificVis 2024].

Comparing the *GD-collection-v1* with the Mooney at al. dataset reveals a distinct metrics landscape: GD drawings come from much smaller graphs (median vertex number 11 vs. 48) and therefore attain higher medians on most readability metrics. For example, Neighbourhood Preservation (0.5 vs. 0.239) is markedly higher in the *GD-collection-v1*. We also find that a large proportion of extracted drawings contain curved and/or polygonal edges (57%), motivating the extended metric definitions.

2012 ACM Subject Classification Human-centered computing → Graph drawings

Keywords and phrases Graph drawing metrics, metric landscape, straight-line drawings, polyline drawings, curved drawings, automated extraction of graph drawings

Digital Object Identifier 10.4230/LIPIcs.GD.2025.30

Supplementary Material *Dataset*: `https://github.com/hegetim/gd-collection` [16]
archived at `swh:1:dir:478a27dd277dc5818bdf699d2a5bc222a010533b`
Software (Source Code): `https://github.com/gavjmooney/geg` [17]
archived at `swh:1:dir:91f45ae7976a74b00a0bf86145b52dd78838fb29`

Funding *Tim Hegemann*: Supported by BMFTR grant 01IS22012C.

Acknowledgements We thank Sebastian Kempf for his contributions to Graph Harvester.

1 Introduction

Graph drawings are typically valued for the extent to which they conform to well-established layout principles (sometimes called "aesthetics" in the literature), for example, minimising the number of edge crossings, minimising the "stress" or maximising the angle between edges incident to the same node. While many of these principles have been used over several years of graph drawing research, they have not all (or always) been defined consistently; for example stress [25]. Recently, Mooney et al. [18] defined ten common principles as "unambiguous metrics" – some defined mathematically, some algorithmically, and all normalised to lie between 0 and 1, where 1 describes the principle in the form that it is assumed best supports human understanding. These metrics apply to connected, straight-line drawings of graphs. The authors also defined a "metrics landscape" that shows the distribution of each of the metric values over almost half a million graph drawings. These graph drawings were created by applying six well-known layout methods (and a random layout) to almost 17,000 randomly generated graphs, five times.

In this work, we generalise the above metrics so that they apply to graph drawings that do not just comprise straight-line edges but also polygonal or curved edges. For two drawings with different metric values showing the same (famous) graph, see Figure 1. This generalisation allows us to explore a wider range of graph drawings, and, arguably, a more representative set of graph drawings. The generalised metrics also allow us to investigate a set of graph drawings that we know are of real interest to the graph drawing community: the drawings in the proceedings of the Graph Drawing conference since 1998.

To this end, we used the Graph Drawing Harvester [5], which was originally designed to extract abstract graphs from a single PDF file (and to check whether the extracted graphs are already contained in the graph database *House of Graphs* [4]). We have extended the Harvester in three directions. Firstly, it can now run in batch, processing many articles one after the other, writing log files with the results. Second, it now checks more carefully for (false) multi-edges and loops. Finally, it now extracts, for each edge, a geometric description; that is, a polygonal line or a Bézier curve that connects its two endpoints. This allows us to apply the (generalised) metrics to the extracted drawings.

(a) Drawing with unit-length straight-line edges (AR 0.24, CA 0.8, EC 0.93, ELD 1, NP 0.09).

(b) Drawing with only two crossings (AR 0.33, CA 0.9, EC 0.97, ELD 0.72, NP 0.4).

Figure 1 Two drawings of the Petersen graph and a selection of their metric values (for the abbreviations and the definition of the metrics, see Section 3.2).

Our contribution. We first present a summary of the existing metrics, and explain how we have extended them to cater for drawings with curves or bends. The Graph Drawing Harvester pipeline and the challenges in its design and implementation are described next. We then present the collection of 4,890 graph drawings that we have extracted from 27 editions of the proceedings of the Symposium of Graph Drawing (and Network Visualization) 1998–2024. To validate this collection, the *GD-collection-v1*, we take a random sample of

extracted drawings from about 10% of the articles and analyse the quality of our extraction, in terms of true/false positives/negatives, and in terms of the number of missing or wrong features of the true positives. We present the metric landscape of the *GD-collection-v1* and compare it to the landscape of generated layouts defined by Mooney et al. [18].

Given the inevitable errors in the completely automated extraction, the *GD-collection-v1* is clearly not a perfect representation of the graph drawings contained in the proceedings. In spite of these errors, we think that this initial version is already a valuable benchmark set of mostly hand-drawn graph drawings that complements the existing benchmark sets (Rome graphs, AT&T graphs) that have been selected in a completely different way and that do not come with an "original" drawing. We will make the extraction code, the *GD-collection-v1*, and the metric code available to the community so that everybody can check the extracted drawings and add missing or remove erroneous features.

2 Background

2.1 Graph Drawing Metrics

To evaluate the aesthetic quality of a graph drawing, researchers define metrics that quantify the extent to which a graph drawing conforms to a particular layout principle. Typical layout principles include minimising edge crossings [21, 23, 24] or maximising symmetry [29].

These metrics are also used in layout algorithms to produce graph drawings by optimising different criteria. For example, Kamada and Kawai [12] explicitly optimise global distance preservation, while Radermacher et al. [24] minimise edge crossings. Graph drawing metrics are also used in human experiments to investigate the perception and task performance of graph layouts [19, 21, 28].

Purchase [22] defines a set of seven metrics, such as *Angular Resolution*, which are normalised to the range $[0, 1]$ to facilitate comparison across drawings of different graphs. Values of 1 represent the extreme that is intuitively assumed to be good, e.g., zero crossings. Mooney et al. [18] extend this set by refining existing metrics and introducing new ones, including the *Gabriel Ratio*. They present the distributions of metric values across different layouts and examine pairwise correlations among metrics. In this paper, we build upon their analysis by comparing these results to those from the *GD-collection-v1*.

The dataset of Mooney et al. [18] comprises 364,094 drawings generated by applying six layout algorithms (each five times) to a set of 16,768 graphs sourced from the North [9] and Rome [6] datasets, as well as from six random graph generators. The layout algorithms include: Fruchterman–Reingold [8], Kamada–Kawai [12], DRGraph [30], Sugiyama et al. [26], Human-like Orthogonal Network Layout (HOLA) [14], and a circular layout.

Due to some layout algorithms being deterministic and bugs in the HOLA implementation, not all combinations yielded unique or valid drawings, hence the dataset size falls short of the expected 503,040 (i.e., $16{,}768 \times 6 \times 5$). To address this, we extend the dataset by reapplying layout algorithms to node-reordered versions of the graphs, thereby generating additional distinct drawings. Despite encountering some further HOLA-related failures, our extended dataset includes 501,185 drawings. We call this the *extended Mooney dataset*.

2.2 Harvesting Graphs

While there are tools for recovering graph structures from either vector [5] or bitmap drawings [2], to the best of our knowledge, there is no tool that allows recovering complex edge geometries such as bends and Bézier curves. In the following, we focus on vector graphics because they are common for manual drawings and easier to process automatically.

Our graph harvesting pipeline consists of three major steps: figure detection, recognising vertex candidates, and edge path recovery. For figure detection, we rely on a modified version of the KIETA pipeline [13] that was originally built to detect tables. It uses keywords to recognise figure captions as well as clustering to find confined areas with many geometric objects. We use the rule-based approach of Deynet et al. [5] to recognise vertex candidates, but we slightly extended the ruleset. Finally, we implemented a greedy algorithm with another set of rules to extract the edge paths. In the process, we need to split each line segment or Bézier curve that contains a vertex in its relative interior.

3 Revised Graph Drawing Metrics

Existing metrics for evaluating graph drawings typically only apply to straight-line, connected, simple graphs. In practice, graph drawings often contain edges with bends/curves, may contain multiple edges between nodes, and comprise disconnected components. This section motivates and describes a new method for representing more complex graph drawings and discusses how the definitions and implementations of existing metrics must be altered to support this representation.

3.1 GEG Encodes Graphs

Existing graph storage formats (such as GraphML [27]) allow edge geometries to be stored as a collection of "bend" points, where these points describe a path of straight-line segments. Extensions to GraphML[1] and other niche formats may allow for edges to be stored as Bézier curves; however, these formats typically do not support (or are not widely used for) more complex edge geometries, which may comprise multiple straight-line and Bézier components. To extract drawings accurately, we must be able to faithfully represent complex edge geometries. We propose a JSON-based format for storing graph drawings: GEG (GEG Encodes Graphs).

GEG is essentially a list of nodes and edges, where each node has an "id" attribute, and each edge has a "source" and "target" attribute, which correspond to node ids. These are the minimum attributes required to describe a graph, but to store a graph drawing, node positions must also be defined. This can be done in two ways: define a "pos" attribute, which is a tuple representing the coordinates of the node, or define "x" and "y" attributes. While node positions are sufficient to encode straight-line drawings, we can also encode complex edge geometries in a "path" attribute, which is formatted as an SVG path command.[2]

This path format was chosen as SVG is a widely deployed format (i.e., with baseline features supported by all major browsers) which is capable of encoding the complex geometries we require. We expect each path to start with an "M" command, which "moves" to the source node's coordinates. This "M" command must be followed by one or more other SVG path commands, such as "L" for a straight-line segment, or "C" for a cubic Bézier curve, which eventually terminate at the coordinates of the target node. If no "path" attribute is included, the edge is assumed to be a straight-line.

A JSON schema for the GEG format and the code to parse GEG files and convert from other common graph formats is included with the metric implementations (see Section 6).

---

[1] `https://www.yworks.com/xml/schema/graphml/1.0/doc/http___www.yworks.com_xml_graphml/element/BezierEdge.html`

[2] Scalable Vector Graphics (SVG) 2 standard. `https://www.w3.org/TR/SVG/paths.html`

3.2 Metric Definitions

The following definitions extend those described by Mooney et al. [18]. We exclude their Gabriel Ratio metric, which is not applicable for drawings with curves, and we include a normalised metric for stress, which is not sensitive to scale and/or rotation [19].

Given a drawing D, let $G(D)$ be the graph that is represented by D, let $V(D)$ be the set of vertices of $G(D)$ (at the positions in D), let $E(D)$ be the set of edges of $G(D)$ (as drawn in D), let $X(D) \subseteq \mathbb{R}^2$ be the set of edge intersection points, and let $\mathcal{X}(D)$ be the set of pairs $(x, \{e, f\})$ that consist of an intersection point x and a pair $\{e, f\}$ of edges that cross at x. For an intersection point x, let $E(x)$ be the set of edges that cross at x.

Angular Resolution (AR) of a drawing D is the deviation of the smallest angle between two edges incident to a vertex compared to the ideal such angle of the same vertex, averaged over the vertices in $V_{>1}(D)$, the set of vertices of degree greater than 1:

$$\mathrm{AR}(D) = 1 - \frac{1}{|V_{>1}(D)|} \sum_{v \in V_{>1}(D)} \left| \frac{\vartheta_v - \vartheta_v^{\min}}{\vartheta_v} \right|, \tag{1}$$

where $\vartheta_v = 360° / \deg(v)$ is the ideal angle at vertex v and $\vartheta_v^{\min}$ is the actual minimum angle between any pair of edges incident to v.

Mooney et al. [18] used the same definition, but our implementation extends theirs; we account for self-loops and multi-edges, and we compute the angle between curved edges at a vertex using the tangent vectors defined by the control points of the Bézier curves.

Aspect Ratio (Asp) of a drawing D is the ratio between the height and width of the axis-aligned bounding box that encloses all vertices and edges. Formally, let $h(D)$ be the height and let $w(D)$ be the width of the bounding box of D. Then

$$\mathrm{Asp}(D) = \begin{cases} 1 & \text{if } h(D) = 0 \text{ or } w(D) = 0, \\ h(D)/w(D) & \text{if } h(D) \leq w(D), \\ w(D)/h(D) & \text{otherwise.} \end{cases}$$

Compared to Mooney et al. [18], our definitions of $h(D)$ and $w(D)$ now take the geometry of the edges into account.

Crossing Angle (CA) of a drawing D is the deviation of the crossing angles from 90°, averaged over every combination of crossing point x and pair of crossing edges crossing at x.

$$\mathrm{CA}(D) = 1 - \frac{1}{|\mathcal{X}(D)|} \sum_{(x,\{e,f\}) \in \mathcal{X}(D)} \left| \frac{\phi_{x,e,f} - \phi_{x,e,f}^{\min}}{\phi_{x,e,f}} \right|, \tag{2}$$

where $\phi_{x,e,f} = 90°$ is the ideal crossing angle of edges e and f at x and $\phi_{x,e,f}^{\min}$ is the smallest angle between edges e and f at x.

To calculate crossings, our implementation splits curved edges into straight-line segments, and checks for crossings between segment pairs. A parameter controls how many segments are used to estimate the curves, which we set to 100. We also include an angular tolerance parameter that specifies the minimum angle where a crossing should be counted (to prevent long overlapping edges being counted as many crossings). We set this to 2.5°. This approach also applies to *Edge Crossings*.

Edge Crossings (EC) of a drawing D counts the number of edge crossings in D, scaled against the total number of potential crossings (in a straight-line drawing of $G(D)$) [22]:

$$\mathrm{EC}(D) = 1 - \begin{cases} c/c_{\max} & \text{if } c_{\max} > 0, \\ 1 & \text{if } c > c_{\max}, \\ 0 & \text{otherwise,} \end{cases} \tag{3}$$

where $c = \sum_{x \in X(D)} |E(x)|^2$ is the number of crossings in D, $c_{\max} = \binom{|E(G)|}{2} - c_{\deg}$ is an upper bound on the number of possible crossings (in a straight-line drawing), and $c_{\deg} = \sum_{v \in V(D)} \binom{\deg(v)}{2}$ is the number of crossings that are impossible due to the fact that adjacent edges cannot cross. A drawing that admits polygonal or curved edges can have more than $c_{\max}$ crossings (as, for example, the assumption that neighbouring edges cannot cross no longer holds). For such a drawing D, we set $\mathrm{EC}(D) = 0$.

Edge Length Deviation (ELD) of a drawing D is the average relative deviation of edge lengths from an ideal value [1]:

$$\mathrm{ELD}(D) = 1 \Bigg/ \left(1 + \frac{1}{|E(D)|} \sum_{e \in E(D)} \left| \frac{L(e) - L_{\mathrm{ideal}}(D)}{L_{\mathrm{ideal}}(D)} \right| \right), \tag{4}$$

where $L(e)$ is the geometric length of edge e, either as the arc length of a curved path or the Euclidean length of a straight-line segment, and $L_{\mathrm{ideal}}(D) = (\sum_{e \in E(D)} L(e))/|E(D)|$ is the average edge length of D. $\mathrm{ELD}(D) = 1$ if and only if all edges of D have the same length.

We refine the metric by explicitly specifying the reciprocal normalisation, thereby improving clarity and interpretability of this metric.

Edge Orthogonality (EO) of a drawing D measures how closely the direction of each edge aligns with the horizontal or vertical axes. For each edge, we compute the angular deviation of its segments from the nearest orthogonal direction (0°, 90°, or 180°), scaled so that 0 indicates perfect alignment and 1 indicates a 45° diagonal. This definition extends that of Mooney et al. [18] to support the inclusion of curved edges.

$$\mathrm{EO}(D) = 1 - \frac{1}{|E(D)|} \sum_{e \in E(D)} \delta_e, \text{ where} \tag{5}$$

$$\delta_e = \sum_{j=1}^{k_e} \frac{\min(\theta_{e,j}, |90° - \theta_{e,j}|, 180° - \theta_{e,j})}{45°} \cdot \frac{\ell_{e,j}}{\tilde{L}(e)} \tag{6}$$

is the length-weighted orthogonality deviation of edge e, each edge e is approximated by a polygonal line $P(e)$ with k_e straight-line segments, $\theta_{e,j}$ is the absolute angle between segment j of $P(e)$ and the horizontal axis, $\ell_{e,j}$ is the length of segment j of $P(e)$, and $\tilde{L}(e) = \sum_{j=1}^{k_e} \ell_{e,j}$ is the total length of $P(e)$.

Kruskal Stress Metric (KSM) is a quality metric adapted from classical multidimensional scaling, originally proposed by Kruskal [15]. It quantifies how well the *relative ordering* of graph-theoretic distances is preserved in the layout, rather than requiring exact spatial accuracy. A more detailed description is given in [25].

To compute the metric, a *Shepard diagram* is constructed by plotting each pairwise Euclidean distance in the layout against the corresponding graph-theoretic (shortest-path) distance. An isotonic regression is then fitted to these points to model the ideal monotonic relationship. The Kruskal Stress Metric of a drawing D is defined as:

$$\mathrm{KSM}(D) = 1 - \sqrt{\frac{\sum_{u,v \in V(D),\, u \neq v} \left(\|u - v\| - \hat{d}_{u,v}\right)^2}{\sum_{u,v \in V(D),\, u \neq v} \|u - v\|^2}} \tag{7}$$

where $\hat{d}_{u,v}$ is the fitted distance from an isotonic regression applied to the Shepard diagram. To maintain consistency with other metrics, we subtract the stress value from 1 (which makes a lower stress correspond to a higher score).

Neighbourhood Preservation (NP) of a drawing D measures how well graph distances (in $G(D)$) between vertices are reflected by their Euclidean distances in D. This is expressed by the Jaccard similarity between the adjacency matrix of $G(D)$ and a geometric k-neighbourhood matrix [1]. Formally,

$$\mathrm{NP}(D) = \frac{|A \wedge M^k|}{|A \vee M^k|}, \tag{8}$$

where A is the adjacency matrix of $G(D)$ (i.e., $A_{uv} = 1$ if vertices u and v are adjacent) and M^k is a binary matrix such that $M^k_{uv} = 1$ if and only if vertex v is one of the k nearest Euclidean neighbours of vertex u in D. We set $k = \lfloor 2|E(D)| \,/\, |V(D)| \rfloor$ to the floor of the average degree.

Node Resolution (NR) of a drawing D is the ratio of the minimum to the maximum Euclidean distances over all pairs of vertices in D. Formally,

$$\mathrm{NR}(D) = \frac{\min_{u,v \in V(D),\, u \neq v} \|u - v\|}{\max_{u,v \in V(D),\, u \neq v} \|u - v\|}. \tag{9}$$

Node Uniformity (NU) of a drawing D measures how evenly vertices are spatially distributed across D. To this end, the bounding box of D is divided into a uniform grid with T (to be determined) rectangular cells, and the deviation between the actual and the ideal number of vertices per cell is used to compute the score. Let n_i denote the number of vertices in cell i, and let $\mu = |V(D)|/T$ be the ideal number of nodes per cell. Then:

$$\mathrm{NU}(D) = 1 - \frac{1}{D_{\max}} \sum_{i=1}^{T} |c_i - \mu|, \tag{10}$$

where $D_{\max} = 2|V(D)|(T-1)/T$ is the worst-case deviation when all nodes are concentrated in a single cell. We set T to match $|V(D)|$ while maintaining roughly square proportions. Specifically, we set the grid dimensions to

$$\#\mathrm{rows} = \left\lfloor \sqrt{|V(D)|} \right\rfloor, \quad \#\mathrm{cols} = \left\lceil \frac{|V(D)|}{\#\mathrm{rows}} \right\rceil, \quad \text{and, hence,} \quad T = \#\mathrm{rows} \cdot \#\mathrm{cols}.$$

If the bounding box of D has height (or width) 0, then $\#\mathrm{rows} = 1$ (or $\#\mathrm{cols} = 1$).

This definition refines the metric proposed by Mooney et al. [18] by improving the handling of cases where the number of vertices is not a square number. Our approach determines the grid dimensions to maintain near-square proportions, while matching the vertex number as closely as possible. Additionally, the normalisation is now based on a well-defined worst-case scenario (all vertices contained in a single cell) rather than relying on an ad-hoc damping mechanism that adjusts scores outside the range $[0, 1]$.

3.3 Directed and Disconnected Graphs

One limitation of these metrics is that they do not consider directed graphs. While they can be applied by converting to undirected graphs (with multi-edges if needed), some layout principles may not translate directly – for example, smaller angles between edges (angular resolution) may be preferable to emphasise directionality.

These metrics can generally be applied to disconnected graphs. For the two that definitionally require connectedness (Kruskal Stress Metric, Neighbourhood Preservation), we apply them to each connected component and return a weighted sum, where weights are based on each component's convex hull area relative to the total.

While the weighted sum approach could be applied to all metrics, it is not always necessary or appropriate. For example, Angular Resolution is locally defined at each node and unaffected by disconnected components. In contrast, using a weighted sum for Edge Crossings would be misleading, as it would ignore crossings between components – an issue if disconnected parts overlap in the drawing. Although such overlap should be avoided to prevent misinterpreting the graph as connected, it may still occur in practice.

4 Extended Graph Harvester

We use and extend Graph Harvester [5], an automated pipeline for extracting vector graph drawings from PDFs. We applied it to the Graph Drawing conference proceedings from 1998–2024. Proceedings from 1997 and earlier are only available as scanned documents and therefore inaccessible to our pipeline that can only handle vector graphics.

4.1 Extraction Logic

The process of harvesting graph drawings can be described by three major phases, namely figure detection, recognising vertex candidates, and edge path recovery. Graph Harvester focuses on the most prevalent type of graph drawings, node-link-diagrams. Here, vertices are represented as individual geometric objects (e.g., small disks) and edges are represented by connecting nodes by straight-lines or curves. We use the term node to refer to the geometric representation of a vertex.

PDFs usually lack semantic information about document structure, so figures must be detected by identifying distinctive patterns (e.g., "Figure" followed by a number and some delimiter) and by analysing areas with many geometric objects and irregular text arrangements. A single figure can contain multiple graph drawings, which can be distinguished by partitioning the figure into non-overlapping clusters of elements.

Naturally, the second step is finding vertex candidates. Usually, all vertices are represented by objects of the same shape and size, but some vertices may use different or larger shapes in order to highlight the vertex or label certain features. Figures also often contain other geometry for annotations that can be mistaken for vertices.

In a third step, we handle edge paths. Ideally, nodes are connected by a distinct sequence of stroked shapes (line segment or Bézier curves) that each start or end at either a node or endpoint of the previous stroked shape in the sequence. This assumption falls short of several patterns used by authors of graph drawings. When drawing self loops, the start and end vertex are identical. Multiedges connect the same pair of nodes via different paths. One entity of geometry may be used to draw multiple edges (e.g., a rectangle that connects four nodes on its corners and therefore represents four edges or a long line segment that

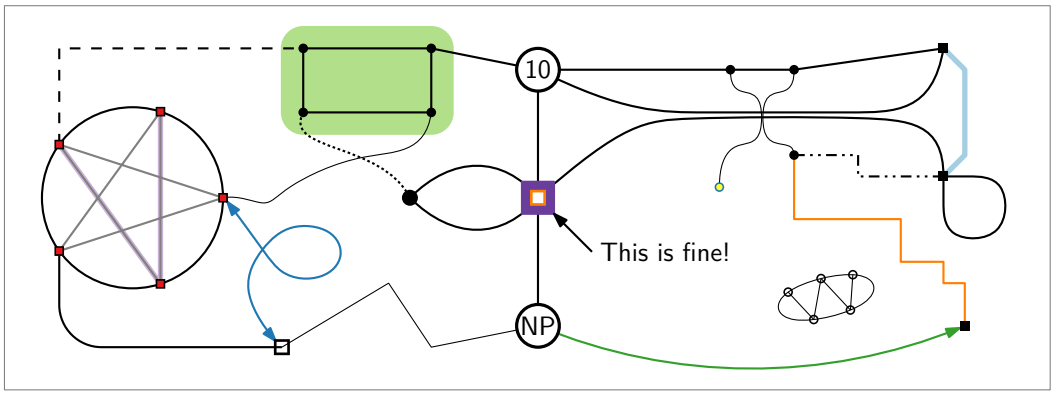

Figure 2 A "test chart" for what the enhanced Graph Harvester can extract. It recognizes all nodes and edge paths; the text labels are filtered out automatically.

connects multiple collinear nodes). Common annotations on edges include arrow heads for expressing edge direction and multi patterning where a thicker shape is placed under the edge for highlighting purposes.

For the Graph Harvester we rely on the following assumptions. Nodes are either squares or circles, they are either stroked (i.e., only the outline is drawn) or filled using a dark colour, and they are clearly separated from each other. Fill may be implemented as multiple stacked shapes of different size (see for example the centre node of Figure 2).

Edges are non-empty sequences of line segments or Bézier curves without jumps (i.e., every segment connects to the next at their respective end/start point), they have a dark stroke, and they connect exactly two nodes (that may be the same node). We recognise edge paths by first splitting any geometry that intersects a vertex candidate, then, matching every endpoint to any overlapping vertex candidate and finally, greedily extending unmatched endpoints with other unmatched pieces of geometry. When extending edge paths, we forbid very acute angles (smaller than 60°) as edges tend to extend towards their destination.

Any remaining geometry (i.e., any path that has an unmatched endpoint) is discarded as annotation as well as all sort of text. Short self loops are also discarded as they usually result from arrow heads or node decorations. We ignore some properties of the geometry such as colour and dash patterns as they are used in many different ways and often highlight vital parts of the drawing. However, we remove very lightly coloured geometry (Rec. 709 luma ≥ 0.75) as these are mainly used for annotation or are not visible at all. After reviewing first results (see Section 4.2), we decided to exclude all drawings where self-loops were detected because true self-loops are very rare in the dataset and the majority of detections stemmed from completely unusable extractions. We also excluded some very large graph drawings for performance reasons.

We composed a graph drawing using a diverse set of styles that match our assumptions, similar to a test chart for imaging systems (see Figure 2). Our extended version of Graph Harvester handles this correctly.

4.2 Evaluation

In order to asses the quality of our graph extraction pipeline, we manually examined 101 papers (about 10%) that we picked at random from our dataset. In these we found 1,358 figures, 1,016 of which were graph drawings (i.e., node-link-diagrams). Note that we counted clearly separated drawings individually even when they were composed into a single figure (e.g., using subfigures). See Figure 3 for an overview.

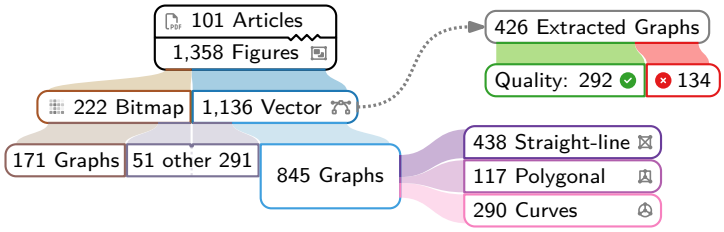

Figure 3 We analysed a sample of 101 articles manually in order to validate our large corpus.

So far, Graph Harvester cannot detect if an edge is directed or not. In our sample, 12% of the graphs were directed (which Graph Harvester can't detect), 11% were multi-graphs (i.e., had multiple edges connecting the same pair of nodes), and 1% had self loops. When looking at the actual drawings, 54% used only straight-line edges, 12% contained at least some polygonal edges, and 34% made use of curved edges. Because our pipeline can only handle vector graphics, we loose all bitmap images in the extraction process. Fortunately, only 17% of the graph drawings were bitmaps.

From the sample, Graph Harvester managed to extract 426 graphs. In 26 cases multiple proximal drawings were detected as one disconnected graph. So the extracted graphs actually stem from 487 individual drawings. For numerous reasons (see Section 4.3 for some examples), 365 graphs could not be extracted. Only seven other figures were incorrectly detected as graphs. For the extracted graphs, we checked how accurate they reproduce the adjacencies and edge paths depicted. An extraction "passes" our assessment if all vertices are recognised and only an insignificant number of edges is missing or misrepresented. In our sample, 69% of the extracted graphs passed resulting in an overall somewhat low rate of 40% high-quality extractions from the vector graph drawings.

Given the very diverse landscape of drawing styles and our generous interpretation of what to count as a graph drawing (see Section 4.3), we still consider these numbers a success. We argue that the quality of our extraction procedure is sufficient to make further assessments based on this dataset.

4.3 Limitations

To explore limitations of our approach, we focus on those graph drawings that caused problems in the extraction process. We compiled the following list of common patterns. Each item refers to the corresponding tile in Figure 4 (a) to (o).

(a) Not a Node-Link-Diagram (but a graph drawing). The figure shows a so-called contact representation. We consider graph drawings that are not node-link-diagrams out of scope (i.e., we did not count them in Section 4.2 and do not expect any extracted graphs).

(b) Not a Graph Drawing (but a node-link-diagram). Some figures that are not intended to show graphs contain geometry that can be mistaken for nodes and edges. We use a set of filter rules in order to prevent such extractions. Few false positives were reported in the evaluation.

(c) Incomplete Drawing. The complete parts are extracted; incomplete edges are discarded.

(d) Disconnected Components and Isolated Vertices. If the drawing is detected as one figure, a disconnected graph is extracted. All isolated vertices are discarded.

(e) Huge and/or Heavily Annotated Nodes. These node types are usually not identified as vertex candidates, so false detections often occurred with such drawings.

(f) Text-only Nodes. No graph is extracted.

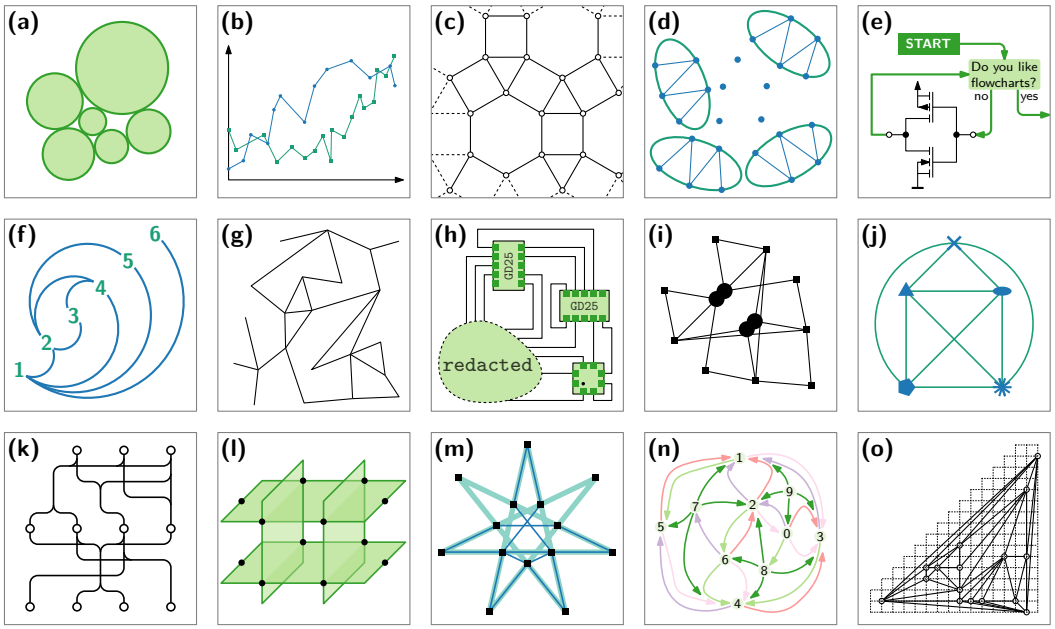

Figure 4 Things we actually found in graph drawings.

(g) No Vertex Markers. No graph is extracted.

(h) Ports or Compose Nodes and Intentionally Obstructed Elements. Nodes of this style are usually not identified as vertex candidates. If geometry is present in the vector data, it is extracted regardless of any patch occluding it.

(i) Partially Overlapping and/or Ambiguously Placed Nodes Partially overlapping nodes are filtered and only one of them is extracted. If a node overlaps an edge, that edge is split into two edges adjacent to that node.

(j) Unusual Node Shapes (i.e., not a square or a circle). No graph is extracted.

(k) Confluent Edges and Hyper Edges (i.e., edges that connect more than two nodes). If each confluent edge consists of an individual geometry object, the extraction succeeds. Otherwise, at most one edge per hyper edge is extracted.

(l) Acute Edge Bends and Filled Edges. Segments resulting in acute angles are not considered for edge continuations. Filled geometry is considered as edge paths.

(m) Two-in-one Drawing. A blend of both graphs is extracted. Multiply exposed edges are filtered and only one copy is extracted.

(n) Lightly Coloured Elements and Edges not Reaching Their Target. Very lightly coloured elements are discarded. Incomplete edge paths are stretched only to a certain degree to prevent false extractions.

(o) Grid or Level Lines. Unfortunately, these are extracted as edge paths.

5 Graph Drawings in the GD Proceedings

5.1 General Analysis

From the 27 proceedings (1998–2024) we extracted 1,037 articles. Of these articles, only 740 (71.4%) contain graph drawings (that we were able to extract). In total we extracted 4,890 graph drawings – 3,769 (77.1%) were connected, 858 (17.6%) were multi-graphs, and 4,211 (86.1%) were planar (though only 2,807 (57.4%) were actually drawn without crossings).

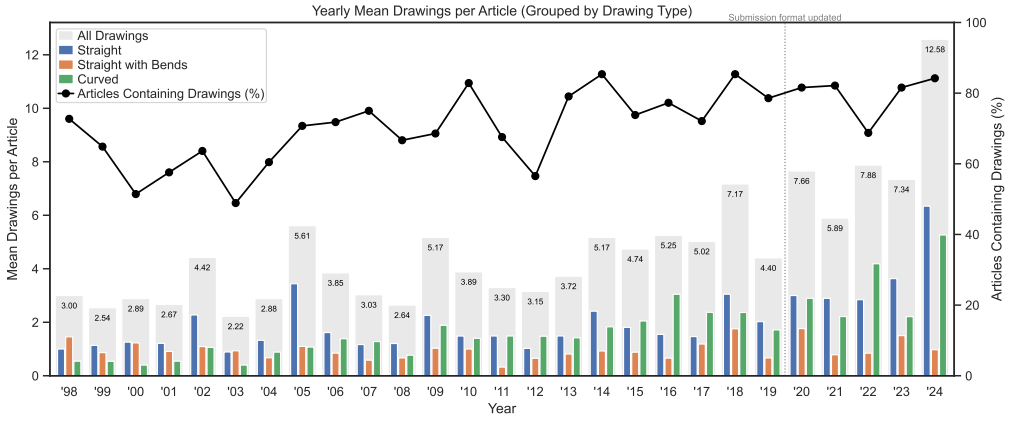

Figure 5 The stacked bar chart shows, for each year, the mean number of drawings per article (grouped by edge style). The line chart shows, for each year, the proportion of articles that contain drawings. In 2020, the submission format for GD was updated to support the inclusion of figures.

We categorise the 4,890 drawings into three sets ("styles") based on edge geometry:

- **Straight**: Those which consist of only straight-line edges: 2,102 (43.0%)
- **Polygonal**: Those which contain polygonal edges (i.e., bends): 998 (20.4%)
- **Curved**: Those which contain curved edges: 1,790 (36.6%)

The significant proportion of drawings which contain curves highlight the importance of defining metrics that can evaluate such drawings.

Figure 5 shows the mean number of drawings per article per year, grouped by drawing type. We fitted linear regressions to examine trends in the number of drawings per article over time. The mean number of drawings per article has increased significantly (slope = 0.22, $R^2 = 0.59$, $p < .001$), as has the proportion of articles that contain drawings (slope = 0.0088, $R^2 = 0.46$, $p < .001$). While these regressions confirm a clear upward trend over the past 27 years, this trend is not expected to continue indefinitely, as there are practical limits to how many drawings can be included in an article. The 2020 change from page limits to line limits – intended to allow more figures – may also have contributed to the observed trend, although our dataset only includes extracted graph drawings (not all figures, or even all drawings). Year-to-year differences also remain substantial, likely due to variations in conference themes, paper content, and editorial guidelines.

Figure 6 shows the distribution of drawings per article, excluding articles with no (extractable) drawings. The median article contains 5 drawings (mean = 6.61). The article with the most drawings (namely 67) was a 2009 paper by J. Joseph Fowler [7] – it contained many small planar trees.

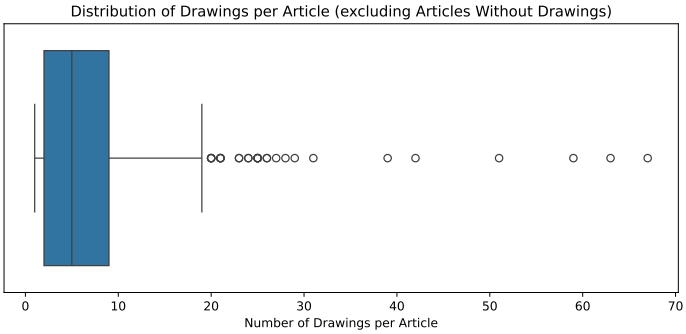

Figure 6 Boxplot of the number of drawings per article, excluding articles with no (extractable) drawings. The median article contains 5 drawings.

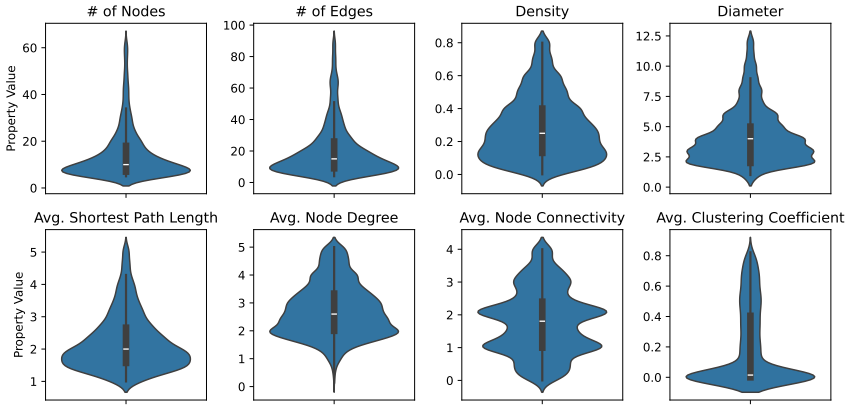

Figure 7 Violin plots showing the distribution of values for 8 common graph properties on the *GD-collection-v1*. Values above the 95th percentile have been excluded to allow for easier inspection of long tail-distributions (such as the number of nodes or edges).

Figure 7 shows the distribution of values for eight common graph properties on our dataset, excluding those above the 95th percentile to allow for easier comparison of long-tail distributions. The complete distributions are shown in Appendix A. All properties were calculated using implementations from the Python library NetworkX [10]. The average node connectivity, defined by Beineke et al. [3], measures the mean number of node-independent paths between all node pairs, reflecting overall network cohesion. The average clustering coefficient, defined by Kaiser [11], captures the tendency of nodes to form local clusters by averaging the proportion of links between each node's neighbours. The density of a graph G is calculated using the standard definition: $d(G) = |E(G)|/\binom{|V(G)|}{2}$. For graphs, density is in $[0, 1]$, but the density of multigraphs may exceed 1.

The majority of extracted graphs have very few nodes (median = 11), and are generally sparse (median density = 0.27). These factors contribute to low diameters (median = 4), average shortest path lengths (median = 2), and average node degrees (median = 2.67). The graphs also tend to have low average node connectivity (median = 1.98), reflecting limited redundancy in paths between node pairs. Average clustering coefficients are similarly low (median = 0.06), suggesting limited local grouping or community structure.

5.2 Metric Analysis of Drawings

In this section we analyse our dataset using the ten readability metrics defined in Section 3.2. Figure 8 shows the value distributions of the ten metrics across our 4,890 extracted drawings (separated by edge type) and the *extended Mooney dataset* (see Section 2.1). Table 1 shows the medians for each metric and dataset (other statistics available in Appendix C).

First, we compare all drawings in the *GD-collection-v1* to the *extended Mooney dataset*. Overall the medians for all metrics are lower in the *extended Mooney dataset* than the *GD-collection-v1*. We attribute this to the greater concentration of larger graphs – the median number of nodes in the *GD-collection-v1* is 11, compared to 48 in the *extended Mooney dataset*. Larger graphs are expected to have poorer performance on most metrics. Full distributions of graph properties for both datasets are provided in Appendix A.

Comparing the three edge geometry types in the *GD-collection-v1* we can also see that distributions are largely similar, though differences highlight some interesting properties. The *Crossing Angle* metric shows a larger distribution of values around 0.7 for the curved drawings when compared to the straight/polygonal drawings. We suggest this stems from arc diagrams and other similar drawings where crossing angle is not optimised.

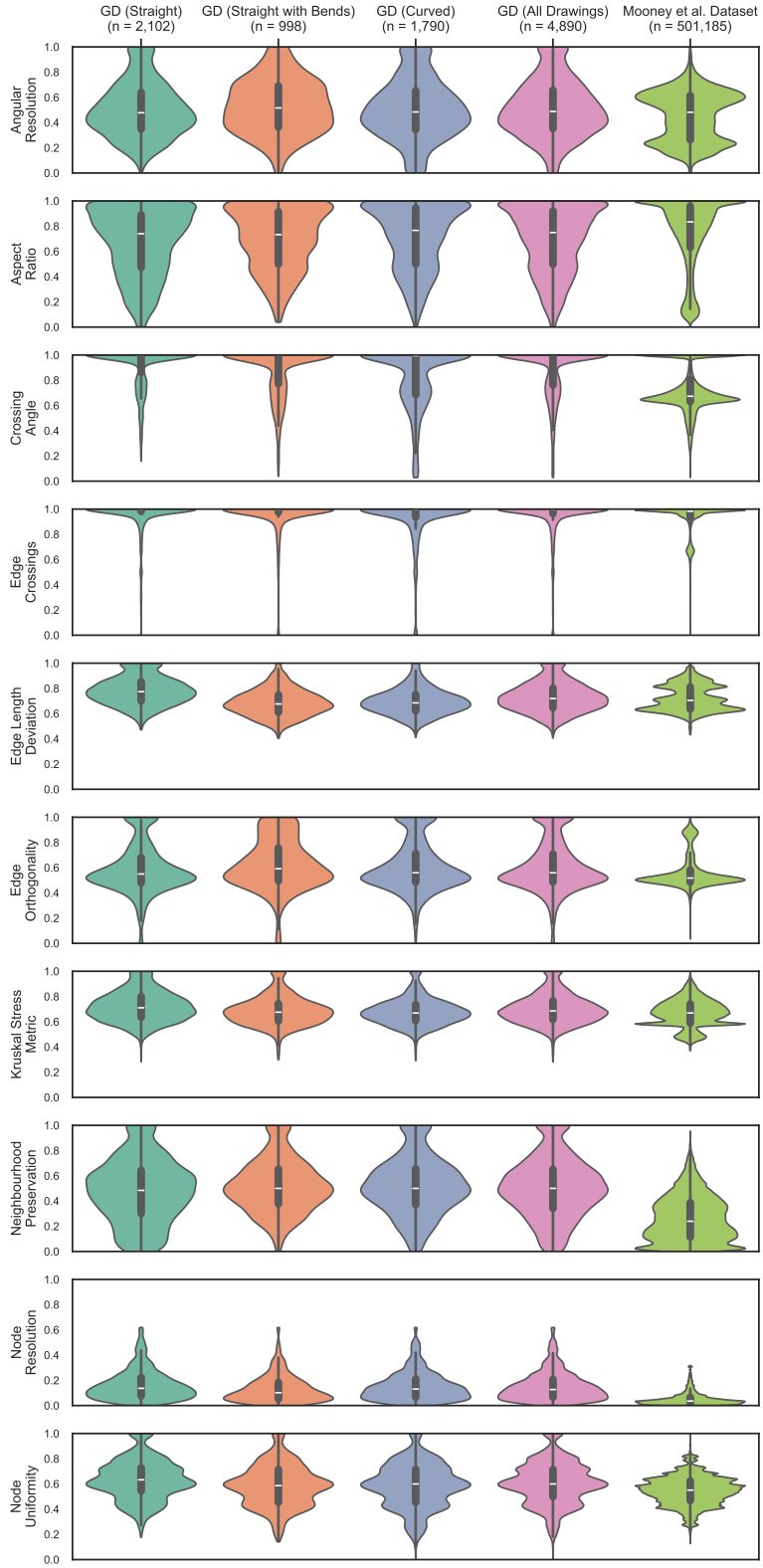

Figure 8 Violin plots showing the distribution of values for 10 metrics on the *GD-collection-v1* (separated by edge type) and the *extended Mooney dataset*.

Table 1 Median for each metric and dataset (see Table 2 in the Appendix for quartiles).

Metric	Straight	Polygonal	Curved	GD Collection v1	Mooney
Angular Resolution	0.478	0.516	0.484	0.488	0.482
Aspect Ratio	0.74	0.733	0.766	0.75	0.834
Crossing Angle	1.0	1.0	0.995	1.0	0.673
Edge Crossings	1.0	1.0	0.998	1.0	0.983
Edge Length Deviation	0.774	0.676	0.685	0.719	0.705
Edge Orthogonality	0.55	0.592	0.56	0.559	0.517
Kruskal Stress Metric	0.71	0.675	0.669	0.686	0.669
Neighbourhood Preservation	0.484	0.5	0.5	0.5	0.239
Node Resolution	0.136	0.101	0.131	0.127	0.036
Node Uniformity	0.633	0.586	0.6	0.6	0.551

Overall, the drawings have very few crossings, but curved drawings tend to admit more crossings than the other two sets (median = 0.999 vs. 1.0). Straight-line drawings clearly perform better on the *Edge Length Deviation* metric, reflecting more consistent edge lengths compared to the variability introduced by curves or bends (median = 0.772 vs. polygonal: 0.677; curved: 0.685). Polygonal-edge drawings perform better for the *Edge Orthogonality* metric (median = 0.589 vs. straight: 0.552; curved: 0.56). We suggest this may be caused by orthogonal layouts which typically contain straight edges with bend points at 90°.

Straight-line drawings perform better on the *Kruskal Stress Metric* (median = 0.709 vs. polygonal: 0.676; curved: 0.669), likely because many stress-minimising layout algorithms are tailored to straight-line representations, even though the metric itself depends only on node positions. Straight-line drawings also perform better on the *Node Uniformity* metric (median = 0.625 vs. polygonal: 0.586; curved: 0.6). Straight-line layouts typically position nodes to reflect structural relationships in the graph, resulting in more uniform spatial distributions. In contrast, bent or curved edge layouts often prioritise edge clarity or visual grouping, which can lead to clustered or uneven node placements and consequently lower node uniformity.

We also examined the pairwise correlations between metrics across both datasets. Overall, the two datasets exhibit generally consistent correlation directions between metric pairs; however, correlation strengths vary. A few discrepancies exist, which we attribute to differences in the types and structures of graphs and layouts included in each dataset. The correlation heatmaps are provided in Appendix B.

6 Conclusion

In this paper, we revised a set of ten quality metrics for evaluating graph drawings, generalising them beyond the traditional constraint of straight-line edges to include drawings with bends and curves.[3] We extended the *Graph Harvester* [5] to obtain an automated pipeline for extracting node-link diagrams – including edge geometries – from vector-based figures in PDFs. Using this tool, we created the *GD-collection-v1*, a dataset of 4,890 graph drawings extracted from 27 years of Graph Drawing conference proceedings.[4] Finally, we analysed this collection and compared its metric landscape to that of an extended benchmark set of over half a million drawings described by Mooney et al. [18].

[3] The implementation of the ten metrics and GEG format is available at `https://github.com/gavjmooney/geg`

[4] The full dataset is available at `https://github.com/hegetim/gd-collection`

Our analysis suggests that while existing readability metrics are broadly applicable, applying them to real-world graph drawings, such as those in the *GD-collection-v1*, reveals subtle differences not evident in datasets comprised solely of synthetically generated layouts. Differences in metric distributions across straight-line, polygonal, and curved drawings indicate that certain drawing styles tend to score better on different graph drawing metrics. The high proportion of curves (37%) and bends (20%) in the *GD-collection-v1* highlights the importance of metric definitions that are applicable to such drawings.

We plan to further grow the GD collection in the future. The community can support this by adhering to some basic guidelines when it comes to graph drawings: Do use vertex markers, use a single shape per edge, and avoid annotations that can be mistaken for edges. In future, more of the existing metrics defined only on straight-line drawings could be extended to work on curved drawings (e.g. those implemented in [20]).

References

1 Reyan Ahmed, Felice De Luca, Sabin Devkota, Stephen G Kobourov, and Mingwei Li. Multicriteria scalable graph drawing via stochastic gradient descent, (SGD)^2. *IEEE Trans. Vis. Comput. Graphics*, 28(6):2388–2399, 2022. doi:10.1109/TVCG.2022.3155564.

2 Christopher Auer, Christian Bachmaier, Franz J. Brandenburg, Andreas Gleißner, and Josef Reislhuber. Optical graph recognition. *J. Graph Algorithms Appl.*, 17(4):541–565, 2013. doi:10.7155/JGAA.00303.

3 Lowell W. Beineke, Ortrud R. Oellermann, and Raymond E. Pippert. The average connectivity of a graph. *Discrete Math.*, 252(1):31–45, 2002. doi:10.1016/S0012-365X(01)00180-7.

4 Kris Coolsaet, Sven D'hondt, and Jan Goedgebeur. House of Graphs 2.0: A database of interesting graphs and more. *Discret. Appl. Math.*, 325:97–107, 2023. doi:10.1016/J.DAM.2022.10.013.

5 Julius Deynet, Tim Hegemann, Sebastian Kempf, and Alexander Wolff. Graph Harvester. In Stefan Felsner and Karsten Klein, editors, *Graph Drawing and Network Visualization (GD)*, volume 320 of *LIPIcs*, pages 58:1–58:3. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. Poster with software presentation. doi:10.4230/LIPIcs.GD.2024.58.

6 Giuseppe Di Battista, Ashim Garg, Giuseppe Liotta, Roberto Tamassia, Emanuele Tassinari, and Francesco Vargiu. An experimental comparison of four graph drawing algorithms. *Comput. Geom.*, 7(5):303–325, 1997. doi:10.1016/S0925-7721(96)00005-3.

7 J. Joseph Fowler. Characterization of unlabeled radial level planar graphs. In David Eppstein and Emden R. Gansner, editors, *Graph Drawing (GD)*, volume 5849 of *LNCS*, pages 81–93. Springer, 2010. doi:10.1007/978-3-642-11805-0_10.

8 Thomas M. J. Fruchterman and Edward M. Reingold. Graph drawing by force-directed placement. *Softw. Pract. Exper.*, 21(11):1129–1164, 1991. doi:10.1002/spe.4380211102.

9 Emden Gansner, Eleftherios Koutsofios, and Stephen North. Drawing graphs with dot. Technical report, AT&T Research, 2006. URL: https://archive.org/details/dotguide.

10 Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart. Exploring network structure, dynamics, and function using NetworkX. In Gaël Varoquaux, Travis Vaught, and Jarrod Millman, editors, *Python in Science Conference (SciPy)*, pages 11–15, 2008.

11 Marcus Kaiser. Mean clustering coefficients: The role of isolated nodes and leafs on clustering measures for small-world networks. *New J. Phys.*, 10(8):083042, 2008. doi:10.1088/1367-2630/10/8/083042.

12 Tomihisa Kamada and Satoru Kawai. An algorithm for drawing general undirected graphs. *Inf. Process. Lett.*, 31(1):7–15, 1989. doi:10.1016/0020-0190(89)90102-6.

13 Sebastian Kempf, Markus Krug, and Frank Puppe. KIETA: key-insight extraction from scientific tables. *Appl. Intell.*, 53(8):9513–9530, 2023. doi:10.1007/S10489-022-03957-8.

14 Steve Kieffer, Tim Dwyer, Kim Marriott, and Michael Wybrow. HOLA: Human-like Orthogonal Network Layout. *IEEE Trans Vis. Comput. Graphics*, 22(1):349–358, 2016. doi:10.1109/TVCG.2015.2467451.

15 Joseph B. Kruskal. Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika*, 29:1–27, 1964. doi:10.1007/BF02289565.

16 Gavin J. Mooney, Tim Hegemann, Alexander Wolff, Michael Wybrow, and Helen C. Purchase. GD-collection-v1. Dataset, swhId: swh:1:dir:478a27dd277dc5818bdf699d2a5bc222a010533b (visited on 2025-11-10). URL: https://github.com/hegetim/gd-collection, doi:10.4230/artifacts.25065.

17 Gavin J. Mooney, Tim Hegemann, Alexander Wolff, Michael Wybrow, and Helen C. Purchase. GEG Encodes Graphs. Software, swhId: swh:1:dir:91f45ae7976a74b00a0bf86145b52dd78838fb29 (visited on 2025-11-10). URL: https://github.com/gavjmooney/geg, doi:10.4230/artifacts.25066.

18 Gavin J. Mooney, Helen C. Purchase, Michael Wybrow, and Stephen G. Kobourov. The multi-dimensional landscape of graph drawing metrics. In *IEEE Pacific Vis. Conf. (PacificVis)*, pages 122–131, 2024. doi:10.1109/PACIFICVIS60374.2024.00022.

19 Gavin J. Mooney, Helen C. Purchase, Michael Wybrow, Stephen G. Kobourov, and Jacob Miller. The perception of stress in graph drawings. In Stefan Felsner and Karsten Klein, editors, *Graph Drawing and Network Visualization (GD)*, volume 320 of *LIPIcs*, pages 21:1–21:17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPIcs.GD.2024.21.

20 Martin Nöllenburg, Sebastian Röder, and Markus Wallinger. GdMetriX – A NetworkX extension for graph drawing metrics. In Stefan Felsner and Karsten Klein, editors, *Graph Drawing & Network Vis. (GD)*, volume 320 of *LIPIcs*, pages 45:1–45:3. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPIcs.GD.2024.45.

21 Helen C. Purchase. Which aesthetic has the greatest effect on human understanding? In Giuseppe Di Battista, editor, *Graph Drawing*, volume 1353 of *LNCS*, pages 248–261. Springer, 1997. doi:10.1007/3-540-63938-1_67.

22 Helen C. Purchase. Metrics for graph drawing aesthetics. *J. Vis. Lang. Comput.*, 13(5):501–516, 2002. doi:10.1006/jvlc.2002.0232.

23 Helen C. Purchase, Robert F. Cohen, and Murray James. Validating graph drawing aesthetics. In G. Goos, J. Hartmanis, J. van Leeuwen, and Franz J. Brandenburg, editors, *Graph Drawing*, volume 1027 of *LNCS*, pages 435–446. Springer, 1996. doi:10.1007/BFb0021827.

24 Marcel Radermacher, Klara Reichard, Ignaz Rutter, and Dorothea Wagner. A geometric heuristic for rectilinear crossing minimization. In *Workshop Algorithm Engin. & Exper. (ALENEX)*, pages 129–138. SIAM, 2018. doi:10.1145/3325861.

25 Kiran Smelser, Jacob Miller, and Stephen Kobourov. "Normalized stress" is not normalized: How to interpret stress correctly. ArXiv preprint, 2024. doi:10.48550/arXiv.2408.07724.

26 Kozo Sugiyama, Shojiro Tagawa, and Mitsuhiko Toda. Methods for visual understanding of hierarchical system structures. *IEEE Trans. Syst. Man Cybern.*, 11:109–125, 1981. doi:10.1109/TSMC.1981.4308636.

27 GraphML Team. The graphml file format, 2002. URL: http://graphml.graphdrawing.org/.

28 Colin Ware, Helen Purchase, Linda Colpoys, and Matthew McGill. Cognitive Measurements of Graph Aesthetics. *Inform. Vis.*, 1(2):103–110, 2002. doi:10.1057/palgrave.ivs.9500013.

29 Erik Welch and Stephen Kobourov. Measuring symmetry in drawings of graphs. *Comput. Graph. Forum*, 36(3):341–351, 2017. doi:10.1111/cgf.13192.

30 Minfeng Zhu, Wei Chen, Yuanzhe Hu, Yuxuan Hou, Liangjun Liu, and Kaiyuan Zhang. DRGraph: An efficient graph layout algorithm for large-scale graphs by dimensionality reduction. *IEEE Trans. Vis. Comput. Graphics*, 27(2):1666–1676, 2021. doi:10.1109/TVCG.2020.3030447.

A Complete Graph Property Distributions

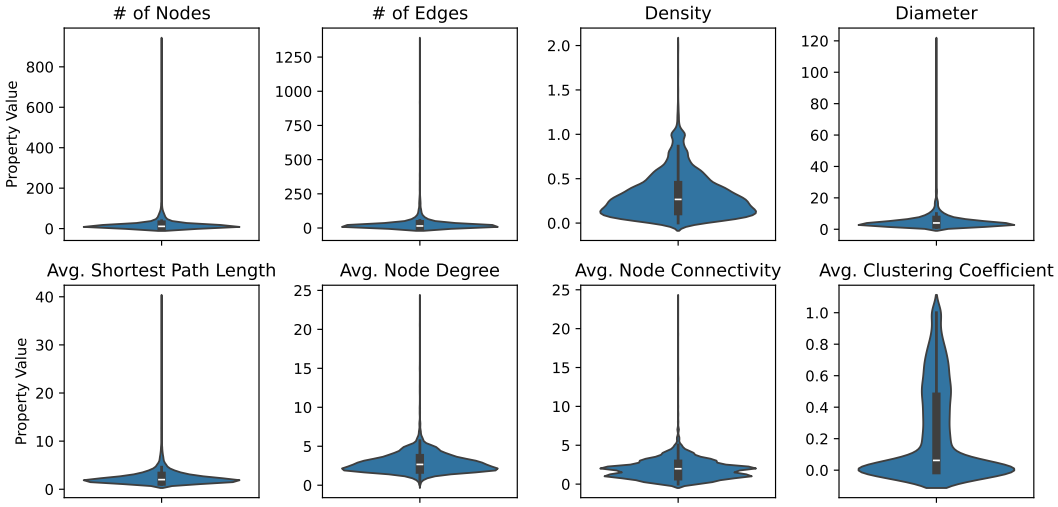

Figure 9 Violin plots showing the distribution of values for 8 common graph properties on the *GD-collection-v1*.

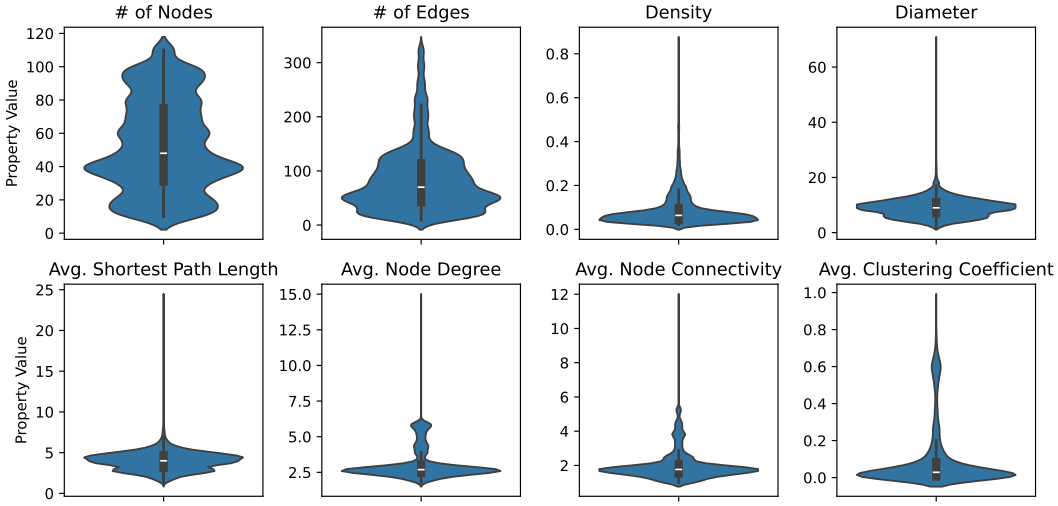

Figure 10 Violin plots showing the distribution of values for 8 common graph properties on the *extended Mooney dataset*.

B Pairwise Metric Correlations

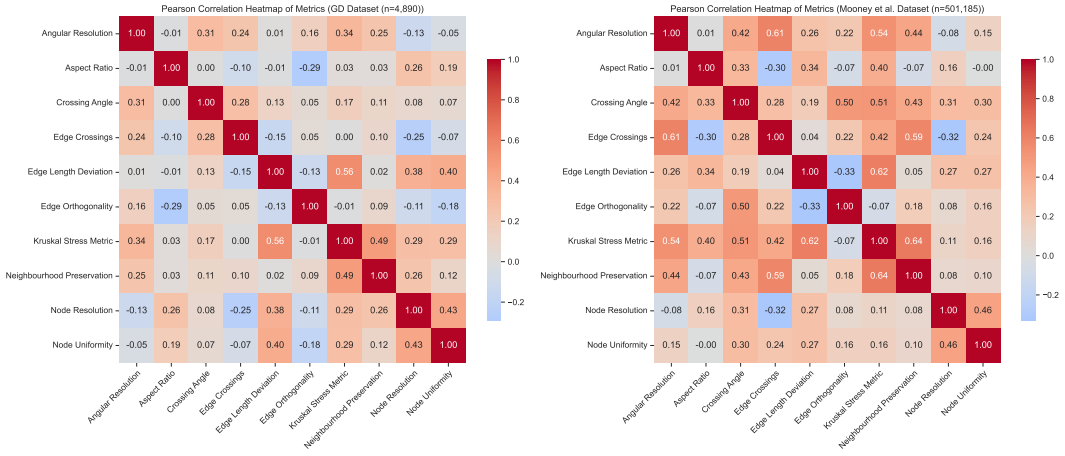

Figure 11 Heatmaps showing the Pearson correlation between pairs of 10 metrics on the *GD-collection-v1* and the extended dataset of 501,185 drawings described by Mooney et al. [18].

C Table of Metric Quartiles

Table 2 Median (M), 1st, and 3rd quartiles for each metric and dataset.

Metric	Stat	Straight	Polygonal	Curved	GD Coll. v1	Mooney
Angular Resolution	Q1	0.348	0.364	0.345	0.352	0.264
	M	0.478	0.516	0.484	0.488	0.482
	Q3	0.639	0.69	0.651	0.657	0.614
Aspect Ratio	Q1	0.474	0.5	0.5	0.499	0.632
	M	0.74	0.733	0.766	0.75	0.834
	Q3	0.889	0.915	0.944	0.92	0.958
Crossing Angle	Q1	0.862	0.773	0.687	0.762	0.631
	M	1.0	1.0	0.995	1.0	0.673
	Q3	1.0	1.0	1.0	1.0	0.807
Edge Crossings	Q1	0.983	0.975	0.937	0.966	0.937
	M	1.0	1.0	0.998	1.0	0.983
	Q3	1.0	1.0	1.0	1.0	0.992
Edge Length Deviation	Q1	0.702	0.611	0.625	0.647	0.637
	M	0.774	0.676	0.685	0.719	0.705
	Q3	0.853	0.749	0.751	0.802	0.812
Edge Orthogonality	Q1	0.48	0.496	0.486	0.486	0.487
	M	0.55	0.592	0.56	0.559	0.517
	Q3	0.68	0.756	0.713	0.708	0.581
Kruskal Stress Metric	Q1	0.636	0.607	0.609	0.618	0.589
	M	0.71	0.675	0.669	0.686	0.669
	Q3	0.798	0.743	0.736	0.766	0.74
Neighbourhood Preservation	Q1	0.3	0.379	0.372	0.341	0.115
	M	0.484	0.5	0.5	0.5	0.239
	Q3	0.643	0.652	0.655	0.65	0.387
Node Resolution	Q1	0.077	0.051	0.067	0.067	0.019
	M	0.136	0.101	0.131	0.127	0.036
	Q3	0.222	0.183	0.209	0.207	0.065
Node Uniformity	Q1	0.545	0.455	0.455	0.5	0.468
	M	0.633	0.586	0.6	0.6	0.551
	Q3	0.727	0.714	0.714	0.714	0.622