Experimental analysis of the accessibility of drawings with few segments

Experimental analysis of the accessibility of drawings with few segments*

Philipp Kindermann[1], Wouter Meulemans[2], and André Schulz[1]

[1] LG Theoretische Informatik, FernUniversität in Hagen
[philipp.kindermann|andre.schulz]@fernuni-hagen.de
[2] Dept. of Mathematics and Computer Science, TU Eindhoven w.meulemans@tue.nl

Abstract. The visual complexity of a graph drawing is defined as the number of geometric objects needed to represent all its edges. In particular, one object may represent multiple edges, e.g., one needs only one line segment to draw two collinear incident edges. We study the question if drawings with few segments have a better aesthetic appeal and help the user to asses the underlying graph. We design an experiment that investigates two different graph types (trees and sparse graphs), three different layout algorithms for trees, and two different layout algorithms for sparse graphs. We asked the users to give an aesthetic ranking on the layouts and to perform a furthest-pair or shortest-path task on the drawings.

1 Introduction

Algorithms for drawing graphs try to optimize (or give a guarantee) on certain formal quality measures. Typical measures include area, grid size, angular resolution, number of crossings, and number of bends. While each of these criteria is well motivated, we have no guarantee that we get a good drawing by optimizing only one of the measures. This is due to the fact that most measures compete with each other. For example, it is known that certain planar graphs cannot be drawn with good angular resolution and polynomial area [8]. The question arises how we can select an appropriate algorithm for a graph drawing task. Instead of relying on a combinatorial or geometric measure of the drawing, one could also value the results of the algorithms by measuring the efficiency of tasks carried out be the observer. Another option would be to just ask the observer which drawing he considers as *"nice"*. By conducting such experiments we also hope to learn something about the formal measures. The goal is here to identify formal measures/algorithms that are particularly suitable for typical tasks performed on graph drawings.

In this paper, we present a study that investigates how good drawings with few segments are perceived by the observer in contrast to other drawing styles.

* This work was partially funded by the German Research Foundation (grant SCHU 2458/4-1). W. Meulemans is funded by the Netherlands eScience Center (NLeSC, grant 027.015.G02).

A drawing with few segments tries to draw several edges that form a path as a single segment. Although the path may contain many edges, this is counted as only one segment in the drawing. The total number of segments is known as the *visual complexity* of the drawing. Instead of straight-line segments, one could also use other geometric objects to draw paths. One option that has been introduced by Schulz [12] is using circular arcs. However, in this paper our focus lies on drawings with segments.

It is an open question whether a small number of segments is a good quality measure for graph drawings. In our study, we want to find out if this design criterion makes drawings more aesthetically appealing for the observer and/or if they are helpful for executing tasks. The main difficulty is that we cannot control the visual complexity of a drawing while keeping other measures fixed. One way to avoid this problem is to adapt existing algorithms in such a way that we can reduce the number of segments in the final drawing without changing too much in the *"style"* of the existing drawing.

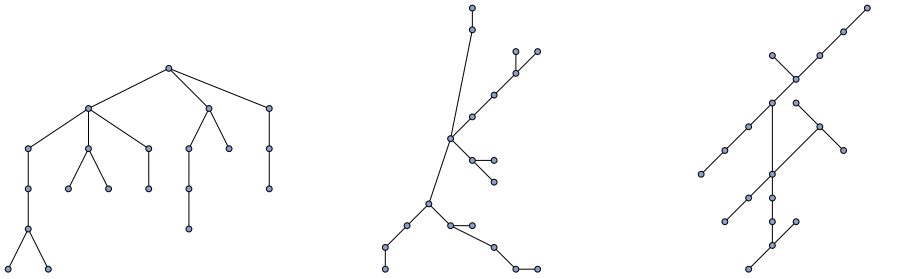
In our study, we focus on two graph classes. The first class are trees, for which many drawing algorithms are known. It is not hard to see that every tree can be drawn with $n_{\text{odd}}/2$ segments, where n_{odd} denotes the number of odd-degree nodes in the tree [3]. It is however unknown if every tree can be drawn with $n_{\text{odd}}/2$ segments using only a polynomial grid size. We heuristically improve the algorithm of Hültenschmidt et al. [6] that draws a tree with minimal visual complexity and quasi-polynomial area and compare its drawing in the user study against drawings of other algorithms. In particular, we use the algorithm of Walker II [14] and of Rusu et al. [11] as alternatives. The former one mimics the standard way how trees are typically drawn in the computer science literature. The latter one aims to draw trees with good angular resolution on a small grid.

The second class of graphs we consider are sparse but not necessarily plane graphs as provided by the *ROME library* [15]. In this setting, it is even harder to control more than one formal measure. We therefore selected only one algorithm to compare with. This is the popular Fruchterman-Reingold spring embedder algorithm [4]. In order to generate drawings that have a *"similar feel"* but use fewer segment, we adapt the spring embedder algorithm by adding constraints that force certain edges to be collinear, and hence form a straight-line segment.

We selected two categories for the users to evaluate the drawings presented to them. The first category addresses the question which of the drawings are aesthetically more appealing to the user. In the second category, we asked the users to perform tasks. For the trees, we asked to identify pair of nodes realizing the largest distance; for the sparse graphs we asked to select a shortest path between two designated vertices. The user study was implemented as a voluntary online questionnaire in order to reach a significant number of participants.

2 Algorithms

Trees For trees, we used three algorithms as illustrated in Fig. 1: `Tidier`, `Quad`, and `FewSegments`. The algorithm `Tidier` was presented by Walker II [14] and



(a) `Tidier` Layout (b) `Quad` Layout (c) `FewSegments` Layout

Fig. 1: A drawing of a tree with each of the three considered layouts.

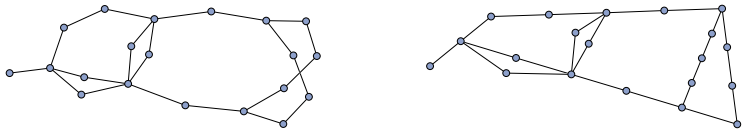
builds upon the classic algorithm by Reingold and Tilford [10]. This algorithm satisfies three criteria: (1) Nodes at the same level of the tree should lie along a straight line, and the straight lines defining the levels should be parallel. (2) A parent should be centered over its offspring. (3) A subtree should be drawn the same way regardless of where it occurs in the tree.

The algorithm `Quad` was presented by Rusu et al. [11]. This algorithm allows the user to specify an angular coefficient and draws edges such that the angles are above the angular coefficient if possible and evenly spread out otherwise. It also allows the user to specify how many quadrants may be used to place the children of a vertex. We chose an angular coefficient of 22.5° and allowed the algorithm to use all four quadrants.

Finally, the algorithm `FewSegments` is based on the algorithm by Hültenschmidt et al. [6] that draws trees on a quasi-polynomial grid with a minimum number of segments. On a high level, that algorithm uses a heavy path decomposition of a tree and places the heavy paths onto segments. It recursively embeds a subtree such that the heavy path of its root is drawn with a vector specified by the parent edge of its root and all subtrees lie in disjoint boxes. We use three heuristics to reduce the size of the drawing.

The first heuristic is applied during the layout of the tree. When the algorithm assigns a vector to a subtree, we allow it to increase the length of the vector slightly such that the new vector is an integer multiple of a smaller primitive vector. For example, if the algorithm would assign a vector $(6, 11)$, then this heuristic would change the vector to $(6, 12)$. This implies that the segments on the heavy path in this subtree do not have to use vectors that are integer multiples of $(6, 11)$, but only integer multiples of $(1, 2)$. Although this makes one segment a bit longer, the subtree might use less area by this change.

The second and the third heuristic are applied in alternating order after a layout has been found. The second heuristic tries to *compress* vectors: given a segment s that is drawn as a vector $\overrightarrow{v}$ that is an integer multiple of a primitive vector $\overrightarrow{u}$, it redraws the tree such that s is drawn with the smallest integer multiple of $\overrightarrow{u}$ without destroying planarity. The third heuristic takes a segment t that is drawn with a long vector $\overrightarrow{w}$ and tries to find a smaller vector $\overrightarrow{w'}$ to draw t



(a) `ForceDir` Layout (b) `FDFewSeg` Layout

Fig. 2: A drawing of the ROME graph *2282.20* with both force-directed layouts.

and the subtree rooted in its child such that the resulting drawing is still planar. This is a more drastic approach and can change the way a subtree is drawn completely. We apply both post-processing heuristics five times on the resulting layout. The second heuristic is applied to all edges, the third only on those whose length is at least 1/5 of the length of the diagonal of the minimum bounding box of the drawing.

Graphs For the sparse graphs, we used the algorithms `ForceDir` and `FDFewSeg`; example drawings are provided in Fig 2. The former is an implementation of the spring embedder by Fruchterman and Reingold [4]. This algorithm computes a force between each pair of vertices. If there is an edge between two vertices, then there is an *attractive force* $f_a(d) = d^2/k$ between them, where d is the distance between the vertices and k is their optimal distance defined as $k = C\sqrt{A/n}$, where C is some constant, A is the maximum area of the drawing, and n is the number of vertices in the graph. If there is no edge between two vertices, then there is a *repulsive force* $f_r(d) = -k^2/d$ between them. By an addition of these forces at every vertex v, we obtain a *movement* Δv of the vertex described by a 2-dimensional vector. Fruchterman and Reingold use simulated annealing to control the movement of the vertices such that the adjustments become smaller over time and the algorithm terminates.

The `FDFewSeg` algorithm is an extension of the `ForceDir` spring embedder. It takes as an additional input a set of edge-disjoint paths $\mathcal{P}$. First, the movement Δv for every vertex is computed. Let $v_0, \dots, v_k = P \in \mathcal{P}$. The algorithm places the vertices $v_1, \dots, v_{k-1}$ evenly spaced onto the segment between v_0 and v_k. To this end, for every vertex $v_i, 0 < i < k$, the movement becomes

$$\Delta v_i = \frac{n-i}{n}\left(v_0 + \Delta v_0\right) + \frac{i}{n}\left(v_k + \Delta v_k\right) - v_i.$$

Note that this procedure does not necessarily draw all paths in $\mathcal{P}$ as segments: if a vertex u of a path is an internal vertex of another path that is processed later, then u will be moved away from its path segment. Hence, the user should input that paths in an order that avoids this problem.

3 Hypotheses

We design a user study to compare aesthetics and legibility of drawings produced by the above-described algorithms. In particular, we pose and analyze for the following four hypotheses:

H1. For trees, the aesthetics ranking is `Tidier` > `FewSegments` > `Quad` for people with mathematics or computer science background and `FewSegments` > `Tidier` > `Quad` for people from a different background.

H2. For trees, path finding is easiest with the `FewSegments` layout, followed by `Tidier`, and hardest with `Quad`.

H3. For sparse graphs, the `ForceDir` layout is more aesthetically pleasing than the `FDFewSeg` layout.

H4. For sparse graphs, path finding is easier with the `FDFewSeg` layout than with the `ForceDir` layout.

Underlying the Hypotheses H2 and H4 is the idea that placing paths onto few segments makes it easier for the user to follow a path between two nodes since the eye only has to move along few directions and can traverse several nodes quickly along a segment. Evenly spacing out the nodes along a path in the force-directed layout should help the reader to quickly determine the number of nodes on a segment and thus to judge the combinatorial length of such a path.

For Hypothesis H1, we guess that the uniformity of the `Tidier` and the `FewSegments` layout would make them win over the `Quad` layout. For mathematicians or computer scientists, we expect that the `Tidier` layout wins since it creates a drawing in the standard way that trees are drawn in the literature. For people with different background, we expect that the `FewSegments` layout wins because it seems to be more schematic.

For Hypothesis H3, we think that the smooth curves in the `ForceDir` layout look nicer to a reader than the drawings of the `FDFewSeg` layouts because the latter ones can have sharp corners at the meeting point of two path segments; for example, Bar and Neta [1] argue that sharp corners have a negative effect on aesthetics as such bends are identified with threat. On the other hand, Vessel and Rubin [13] studied the objectiveness of taste–their conclusion is that there is typically agreement for natural images, abstract depictions are influenced more by individual taste. Though they cannot fully be eliminated, we believe that the uniformity of graphical presentation may mitigate personal preferences to allow for investigating an overall agreement in aesthetics.

4 Experimental design

Selecting tasks We used two tasks: **Aesthetics** and **Query**. We created different graphs for each task. For the Aesthetics task, we showed the user one drawing for each layout of the same graph next to each other. The order of the drawings was determined randomly. The user was asked to determine a ranking on the aesthetics of the drawings by clicking on them in the desired order.

We used different Query tasks based on the graph class. We showed the user one drawing at a time. Over time, every graph was presented the user once with each layout.

For the sparse graphs, we asked the users to find the shortest path between two randomly marked vertices that have distance at least 3 (the pair of vertices was the same for each layout and each user). The user solved this task by clicking on the vertices (or edges) in the order that they appear on this path. To make sure that a user does not get stuck on a question, we allowed them to submit their answer even if no path was found. We helped the user with this task by marking (in a different color) the valid nodes and edges they can click on, which are those that are adjacent to the endpoint of one of the two paths starting in the two marked vertices.

For trees, shortest paths are uniquely defined which makes it unsuitable as a task. Hence, we asked the user to find the furthest pair of vertices, that is, the pair of vertices such that the distance between them is maximized. This also requires the user to inspect several paths in the graph. The user then had to click on the vertices that they determined as the furthest pair.

Generating stimuli For trees, we have the following two variables for the stimuli:

- **Size.** Two different sizes: (1) 20 nodes and (2) 40 nodes.
- **Depth.** Three different tree depths as defined by the length of the longest root–leaf paths: (D) *deep* trees of depth 8 for size 1 and of depth 14 for size 2, (B) *balanced* trees of depth 5 for size 1 and of depth 9 for size 2, and (W) *wide* trees of depth 3 for size 1 and of depth 5 for size 2.

To construct random trees of given size and depth, we create a uniformly distributed random Prüfer sequence [9] and check whether the corresponding labeled tree has the given depth. It is known that Prüfer sequences provide a bijection between the set of labeled trees on n vertices and the set of sequences of $n-2$ integers between 1 and n. Hence, this algorithm gives us uniformly distributed random trees of a given depth. For each size and depth, we created four different graphs for the Aesthetics task and two different graphs for the Query task. This gives us $2 \cdot 3 \cdot 4 = 24$ graphs for the Aesthetic tasks (4 repetitions) and $2 \cdot 3 \cdot 2 = 12$ graphs for the Query task (2 repetitions).

For the sparse graphs, we have the following two variables for the stimuli:

- **Size**. Two different sizes: (1) 30 nodes and (2) 60 nodes.
- **Type**. Two different types: (A) graphs from the ROME library and (B) random graphs.

For graphs of type A, we randomly picked graphs of the given size from the ROME library [15] that consists of 11,535 sparse, but not necessarily planar, graphs with 10 to 100 vertices. For graphs of type B, we created a random graph by creating a number of nodes specified by the size and picking 30 random edges for graphs of size 1 and 60 random edges for graphs of size 2; we used the resulting graph if and only if it is connected. For each size and type, we again created four different graphs for the Aesthetics task and two different graphs for the Query task. This gives us $2 \cdot 2 \cdot 4 = 16$ graphs for the Aesthetic tasks (4 repetitions) and $2 \cdot 2 \cdot 2 = 8$ graphs for the Query task (2 repetitions).

We created the graphs as JSON files that contained the coordinates of the vertices and the set of edges. During the study, the graphs were drawn using the JavaScript library *D3.js* [2] as SVG figures to allow arbitrary resizing. The nodes were drawn using blue circles. Links were drawn in black with a small halo to increase separability between crossing links. The selected vertices and links in both Query tasks were marked in green and the selectable vertices and links in the shortest path task were marked in light blue.

Further considerations For trees, we created 24 stimuli for the Aesthetics task and 36 stimuli for the Query task (one per graph and layout). For the sparse graphs, we created 16 stimuli for the Aesthetics task and 16 for the Query task. This gives us 92 stimuli in total. This is beyond what is reasonable for an online study, assuming 15 to 25 seconds per trial. Since the study has two different graph classes with different tasks, we used the graph class as a between-subjects measure. This still leaves 60 stimuli for the tree tasks. Since the size of a graph is very likely to be an overall factor by the larger difficulty of the Query task on a larger graph, we used the size as an additional between-subjects measure for trees. This way, we obtain three groups of stimuli: (1) 30 stimuli for trees of size 1, (2) 30 stimuli for trees of size 2, and (3) 32 stimuli for sparse graphs. A pilot study showed a completion time of about 15 minutes for each group.

We first show the Aesthetics task and then the Query task. We did this such that the user does not get a bias for a specific drawing style based on the difficulty of the Query task and instead of the most aesthetic one picks the one that they preferred in the Query section. Though explicitly asking for visual preference could bias performance in the following Query section, we expect this effect to be negligible as only one drawing is shown at any given time; and in any case less strong than the potential bias if the sections were to be inverted.

In order to account for learning effects, the order of the stimuli for each task was randomized for each participant. Before each stimulus, the participant was given a pause screen to reduce memory effects and at the same time allow them to pace themselves and reduce the possible impact of interruptions. The participants received one example question with an answer revealed after providing one, from which they could go back to task description, to ensure that the task was understood before starting the actual questions. We opted not to provide a longer series of training questions to keep time investment to a minimum.

Setup We developed our user study with PHP and the JavaScript library *D3.js*. The study was hosted on a web server[3] and the data was stored in a MySQL database. Since the questionnaire was conducted online, we had no control over many parts of the experimental environment, e.g., device, pointing device, operating system, browser, screen size, interruptions. We asked the participants to fill in the questionnaire using a desktop or laptop computer, not a tablet or phone, and to use the pointing device they are most comfortable with. To make sure that the browser is suitable to run the questionnaire, the users first had to

[3] http://tutte.fernuni-hagen.de/web/userstudy/fewarcs

set a slider to the value depicted in an SVG figure. We could not control the screen size, resolution, or distance of the participant to their screen, so we let the user control the scale of the web page by providing a *Shrink* and a *Grow* button. Further, we asked them to put their browser in full-screen mode to reduce distractions. We requested the participants to not engage in other activities during the questionnaire and to minimize interruptions, and to specify if any interruptions occurred at the end of the study.

We recruited the voluntary participants of the user study using a mix of mailing lists, social networks, and social media. Some background and preference information was asked upon completion, although this remained optional for what may considered sensitive information (age, gender, country of residence).

5 Results

The data set for the analysis as well as all stimuli have been made available online[4]. In total, 84 people volunteered and completed the online questionnaire, which was open for participation for two weeks. We inspected all comments left by participants. One participant had a longer break during one of the questions, rendering this particular question unsuitable for the analysis. As to maintain a balanced design to allow for stronger analysis methods, we excluded him from the analysis. This gave us 21 participants for both group 1 and group 2, and 41 participants for group 3. Of the 83 participants, 75 provided their age with an average of 36.96. In terms of country of residence, a majority of the participants live in Europe (63), predominantly in Germany (42).

Hypothesis H1 For the tree aesthetics task, we had 42 participants from groups 1 and 2 and each of them was shown 12 stimuli. This gave us a total of 504 rankings between the three layouts. We used loglinear Bradley-Terry (LLBT) modeling [5] of the 1,512 pairwise aesthetic preference comparisons to produce ranked *worth* scores for each of the three layouts. The worth score allows the consistency of preference to be assessed in forming an overall ranking of the three classes. Fig. 3 shows the ranking of the three layouts in terms of aesthetic preference, broken down by the balance of the graph and by the background of the participants.

Consistently, the `Quad` layout was considered as the least aesthetic tree layout. Over all answers, the `Tidier` layout performed the best. There was some effect based on the balance of the graph. For each balance, we received 168 rankings. For *balanced* trees, the layouts `FewSegments` (worth score 0.4308) and `Tidier` (worth score 0.4302) were perceived equally aesthetic. However, for *deep* and *wide* trees the `Tidier` layout performed better than `FewSegments` with worth scores of 0.4757 versus 0.3785 (deep) and 0.4881 versus 0.3959 (wide).

The hypothesis was split into two parts, depending on the background of the participants. Let us first consider the participants with a mathematics or computer science background. There were 48 rankings by four people with a mathematics background, but not computer science; for those, the `Tidier` layout

[4] http://tutte.fernuni-hagen.de/web/userstudy/fewarcs/studyresults.html

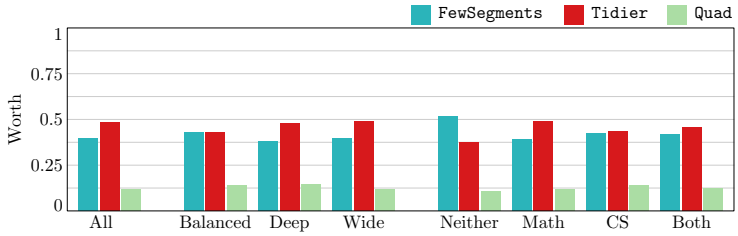


Fig. 3: Worth scores of the three tree layout methods: overall and partitioned by tree balance or participant background.

was clearly preferred with a worth score of 0.4874 over 0.3929. There were 312 rankings by participants with a computer science background, but not mathematics; for those, the layouts were perceived similarly with a worth score of 0.4360 for `Tidier` and 0.4251 for `FewSegments`. There were 120 rankings by users with both mathematics and computer science background; those slightly preferred the `Tidier` layout with a worth score of 0.4559 over 0.4196. Overall, this suggests that there is some preference of the `Tidier` layout over the `FewSegment` layout, and of both layouts over the `Quad` layout; hence, we tentatively accept the first part of Hypothesis H1, due to the caveat of a small number of participants with a math-only background.

The second part of Hypothesis H1 is about participants from neither mathematics nor computer science background. There is some evidence for the hypothesis to be true. The participants from neither background strongly preferred the `FewSegments` layout over the `Tidier` layout with a worth score of 0.5173 over 0.3761. However, this only included 24 rankings from 2 participants, and thus the findings are a suggestion at best. This also indicates that our method for recruitment was not sufficiently broad enough to recruit volunteers from a variety of backgrounds. Indeed, this affects the other of the results as well.

Hypothesis H2 For the tree query task, we had 42 participants from groups 1 and 2 and each of them was shown 18 stimuli. This gave us a total of 756 tasks between the three layouts with 252 tasks per layout. We analyzed the error rates for finding a furthest pair for the three tree layouts defined by the difference of the distance between the picked pair and the distance between a furthest pair in the graph, broken down by the balance and by the size of the trees. The maximum response time was 53 seconds, so we did not have to exclude any participants. Fig. 4 shows the error rates and the answer times by the participants.

We used a two-way RM-ANOVA to analyze the effects of the layouts, tree balance, tree size, and their interaction. We used the logarithm of the response times to normalize the distribution. For the error rate, there are no interaction effects between layout, balance, and size. The analysis showed a weak effect of the layout on the error rate ($F(2, 80) = 3.636$, $p < 0.05$). A post-hoc Tukey HSD test with Bonferroni adjustment showed a significant difference between layout `Quad` and `FewSegments` in favor of `FewSegments` ($p < 0.01$) and a signifi-

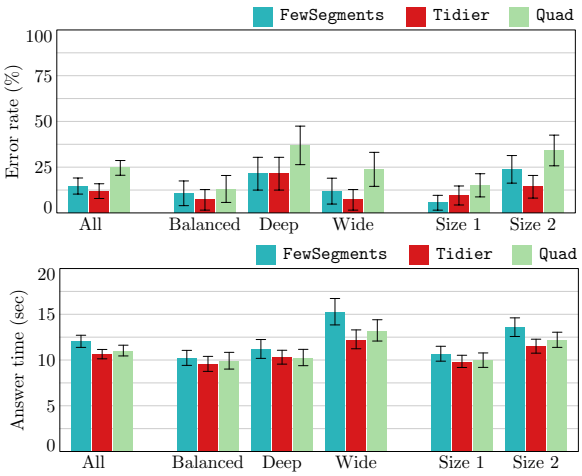


Fig. 4: Error rates and answer times for finding a furthest pair for the three tree layout methods: overall and partitioned by tree balance or size group. Error bars indicate 95% confidence intervals.

cant difference between layout `Quad` and `Tidier` in favor of `Tidier` ($p < 0.001$), but no significant difference between layout `Tidier` and `FewSegments`. Further, a post-hoc test showed a weak difference between the tree sizes ($p < 0.05$) in favor of smaller trees. For small trees, there is some evidence that `FewSegments` outperforms `Tidier` ($p < 0.05$); for large trees the error rate seems lower for `Tidier`, though no statistically significant effect was found ($p > 0.15$). We conclude that the layouts `FewSegments` and `Tidier` perform better than the layout `Quad`, while the participants performed better on small trees than on large trees.

For the response time, there is some interaction between tree size and tree balance ($F(4, 160) = 2.524$, $p < 0.05$), so we split according to sizegroup for further analysis. For small trees, there are no interaction effects between layout and tree balance. The analysis showed a very weak effect of layout ($F(2, 40) = 2.523$, $p < 0.1$) on response time. A post-hoc test showed a very weak difference between layout `Tidier` and `FewSegments` in favor of `Tidier` ($p < 0.1$) and no significant difference between the other two layout pairs. We conclude that the participants performed slightly faster for the `Tidier` layout than for the `FewSegments` layout

For large trees, there are also no interaction effects between layout and tree balance. The analysis showed significant effect of layout ($F(2, 40) = 9.667$, $p < 0.001$) on response time. A post-hoc test showed a weak difference between layout `Quad` and `FewSegments` in favor of `Quad` ($p < 0.05$) and a significant difference between layout `Tidier` and `FewSegments` in favor of `Tidier` ($p < 0.001$). We conclude that the participants performed slightly faster for the `Quad` layout than for the `FewSegments` layout and significantly faster for the `Tidier` layout than for the `FewSegments` layout.

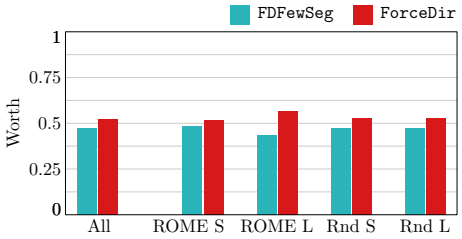


Fig. 5: Worth scores of the two layout methods: overall and partitioned by type.

Since the error rate was smaller for the `FewSegments` and `Tidier` layouts than for the `Quad` layout, but the response time for `FewSegments` was worse than for the other two, we can only partly accept Hypothesis H2: the layouts `FewSegments` and `Tidier` both outperform the layout `Quad`, but the layout `Tidier` outperforms the layout `FewSegments`. Though not initially hypothesized, we also found evidence of an effect of the tree balance on both the error rate and the response time (see the full version [7]) in favor of *balanced* and *wide*.

Hypothesis H3 For the sparse graph aesthetics task, we had 41 participants from group 3 and each of them was shown 16 stimuli. This gave us a total of 656 rankings between the two layouts. We again used LLBT modeling of the 656 pairwise aesthetic preference comparisons to produce ranked worth scores for both layouts. Fig. 5 shows the ranking of the both layouts in terms of aesthetic preference, broken down by the graph class and the size of the graph.

Over all 656 rankings, the `ForceDir` layout was preferred with a worth score of 0.5246 over the `FDFewSeg` layout with a worth score of 0.4754. The `ForceDir` layout was preferred for each pair of graph class and size. The preference is the smallest for small graphs of the ROME library with a difference in worth score of 0.0316, and it is the largest for large graphs of the ROME library with a difference in worth score of 0.1267. Hence, we accept Hypothesis H3.

Hypothesis H4 For the sparse graph query task, we had 41 participants from group 3 and each of them was shown 16 stimuli. This gave us a total of 656 tasks between the two layouts with 328 tasks per layout. We analyzed the error rates for finding a shortest path for the two layouts defined by the difference between the length of the selected path and the length of a shortest path, broken down by the four graph types (ROME small, ROME large, Random small, Random large). Fig. 6 shows the error rates and answer times by the participants.

We used the same analysis as for the tree query task. For the error rate, there is some interaction between layout and graph type ($F(3, 120) = 3.313$, $p < 0.05$), so we split according to graph type. For *Random small* graphs, we found a significant difference between the layouts in favor of `ForceDir` ($F(1, 40) = 9.949$, $p < 0.01$); for the other graph types, there is no significant effect of the layouts.

For the response time, there is a strong interaction between layout and graph type ($F(3, 120) = 21.06$), $p < 0.001$), so we split according to graph type. For

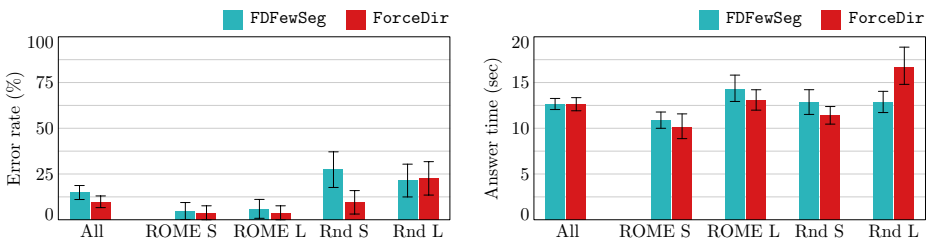


Fig. 6: Error rates and response times for finding a shortest path for the two layout methods: overall and partitioned by graph type. Error bars indicate 95% confidence intervals.

ROME small ($F(1, 40) = 9.317$, $p < 0.01$) and *Random small* graphs ($F(1, 40) = 7.474$, $p < 0.01$), there is a significant effect of the layouts on the response time in favor of `ForceDir`. For *ROME large* graphs, there is a very weak effect of the layouts on the response time in favor of `ForceDir` ($F(1, 40) = 3.901$, $p < 0.1$). For *Random large* graphs, there is a significant effect of the layouts on the response time in favor of `FDFewSeg` ($F(1, 40) = 24.56$, $p < 0.001$).

Since the `ForceDir` layout outperformed the `FDFewSeg` on three of the four graph layouts, we have to reject Hypothesis H4 in general. However, the `FDFewSeg` performed better on large random graphs, so there is some evidence that this layout can give better results if the input graph has many vertices. The reason for this may be that the ROME graphs tend to have many degree-2 vertices. Similar effects can be observed for the small random graphs. Consequently, paths become easily traceable for these instances even if drawn with many segments.

6 Conclusion

We compared various graph layout algorithms to assess the effect of low visual complexity on aesthetics and performance. We have partially confirmed Hypothesis H1, that is, that for trees people with a math or computer science background tend to prefer the classical top-down layout. We also found some evidence that people with no such background prefer the layouts produced by the algorithm assuring low visual complexity; however, lacking a large number of participants in this category makes this only a suggestion of a possible effect. We have also partially confirmed Hypothesis H2, by finding evidence that finding a furthest pair is the easiest with the classical tree layout. We accepted Hypothesis H3 that for sparse graph the traditional force-directed layout is more aesthetic than its modification to reduce the visual complexity. We rejected Hypothesis H4 in general, but rather found that it is typically easier to find the shortest paths between two nodes with the traditional force-directed layout than the modification, though our hypothesis was found to hold for large random graphs. This leaves the possibility open that for graphs that are more intertwined using few segments can be beneficial.

In short, our findings suggest that visual complexity may positively influence aesthetics, depending on the background of the observer, as long as it does not introduce unnecessarily sharp corners. Hence, drawings trees with few segments give a more schematic alternative over the classic drawing style without the risk of harming the aesthetic perception. However, few-segment drawings tend not to improve task performance. It is worth noting that we did not provide training to our participants, as to suggest how the segments can for example help to easily assess the length of a subpath. Providing such clues may have a positive effect on the performance, but at the same time would also result in an unfair comparison, if no training or strategies for the traditional layout were to be suggested.

Acknowledgments The authors would like to thank all anonymous volunteers who participated in the presented user study.

References

1. M. Bar and M. Neta. Humans prefer curved visual objects. *Psychological Science*, 17(8):645–648, 2006.
2. M. Bostock, V. Ogievetsky, and J. Heer. D^3 data-driven documents. *IEEE Trans. Visual. Comput. Graphics*, 17(12):2301–2309, 2011.
3. V. Dujmović, D. Eppstein, M. Suderman, and D. R. Wood. Drawings of planar graphs with few slopes and segments. *Comput. Geom.*, 38(3):194–212, 2007.
4. T. M. J. Fruchterman and E. M. Reingold. Graph drawing by force-directed placement. *Softw., Pract. Exper.*, 21(11):1129–1164, 1991.
5. R. Hatzinger and R. Dirrich. prefmod: An R package for modeling preferences based on paired comparisons, rankings, or ratings. *J. Statist. Softw.*, 48(10):1–31, 2011.
6. G. Hültenschmidt, P. Kindermann, W. Meulemans, and A. Schulz. Drawing trees and triangulations with few geometric primitives. In H. L. Bodlaender and G. J. Woeginger, editors, *Proc. 43rd Int. Workshop Graph-Theor. Concepts Comput. Sci. (WG'17)*, volume 10520 of *Lecture Notes Comput. Sci.* Springer, 2017. To appear.
7. P. Kindermann, W. Meulemans, and A. Schulz. Experimental analysis of the accessibility of drawings with few segments. *Arxiv report 1708.09815*, 2017.
8. S. M. Malitz and A. Papakostas. On the angular resolution of planar graphs. *SIAM J. Discrete Math.*, 7(2):172–183, 1994.
9. H. Prüfer. Neuer Beweis eines Satzes über Permutationen. *Arch. Math. Phys.*, 27:742–744, 1918.
10. E. M. Reingold and J. S. Tilford. Tidier drawings of trees. *IEEE Trans. Software Eng.*, 7(2):223–228, 1981.
11. A. Rusu, C. Yao, and A. Crowell. A planar straight-line grid drawing algorithm for high degree general trees with user-specified angular coefficient. In *Proc. 12th Int. Conf. Inform. Vis. (IV'08)*, pages 600–609. IEEE Computer Society, 2008.
12. A. Schulz. Drawing graphs with few arcs. *J. Graph Algorithms Appl.*, 19(1):393–412, 2015.
13. E. Vessel and N. Rubin. Beauty and the beholder: Highly individual taste for abstract, but not real-world images. *Journal of Vision*, 10(2):1–14, 2010.
14. J. Q. Walker II. A node-positioning algorithm for general trees. *Softw., Pract. Exper.*, 20(7):685–705, 1990.

15. E. Welzl, G. Di Battista, A. Garg, G. Liotta, R. Tamassia, E. Tassinari, and F. Vargiu. An experimental comparison of four graph drawing algorithms. *Comput. Geom.*, 7:303–325, 1997.