OXFORD UNIVERSITY PRESS | The Computer Journal

A Heuristic Approach towards Drawings of Graphs with High Crossing Resolution

Journal:	*The Computer Journal*
Manuscript ID	Draft
Manuscript Type:	Original Article
Date Submitted by the Author:	n/a
Complete List of Authors:	Bekos, Michael; Universität Tübingen Förster, Henry; Universität Tübingen Geckeler, Christian; Universität Tübingen Holländer, Lukas; Universität Tübingen Kaufmann, Michael; University of Tuebingen, Institute for Informatics Spallek, Amadäus; Universität Tübingen Splett, Jan; Universität Tübingen
Key Words:	Graph Drawing, Crossing resolution, Angular resolution, Total resolution, Heuristic algorithm

A Heuristic Approach towards Drawings of Graphs with High Crossing Resolution

Michael A. Bekos, Henry Förster, Christian Geckeler, Lukas Holländer, Michael Kaufmann, Amadäus M. Spallek and Jan Splett

Institut für Informatik, Universität Tübingen, Tübingen, Germany
Email: {bekos,foersth,mk}@informatik.uni-tuebingen.de
{geckeler,jan-lukas.hollaender,amadaeus.spallek,jan.splett}@student.uni-tuebingen.de

The *crossing resolution* of a non-planar drawing of a graph is the value of the minimum angle formed by any pair of crossing edges. Recent experiments suggest that the larger the crossing resolution is, the easier it is to read and interpret a drawing of a graph. However, maximizing the crossing resolution turns out to be an NP-hard problem in general, and only heuristic algorithms are known that are mainly based on appropriately adjusting force-directed algorithms.
In this paper, we propose a new heuristic algorithm for the crossing resolution maximization problem and we experimentally compare it against the known approaches from the literature. Our experimental evaluation indicates that the new heuristic produces drawings with better crossing resolution, but this comes at the cost of slightly higher aspect ratio, especially when the input graph is large.

Keywords: Graph Drawing, Angular, Crossing and Total Resolution.

Received 13 September 2018

1. INTRODUCTION

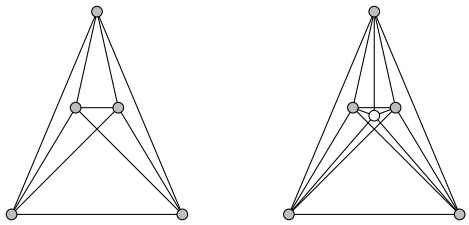
In Graph Drawing, there exists a rich literature and a wide range of techniques for drawing planar graphs; see, e.g., [1, 2, 3]. However, drawing a non-planar graph, and in particular when it does not have some special structure (e.g., degree restriction), is a quite difficult and very challenging task, mainly due to the edge crossings that negatively affect the drawing's quality [4]. As a result, the established techniques are significantly fewer (e.g., crossing minimization heuristics [5, 6], energy-based layout algorithms [7, 8]); for an overview we point the reader to [9, 10, 11].

In this context, Huang et al. [12, 13] a decade ago introduced some important experimental evidence (through eye-tracking experiments), that edge crossings may not negatively affect the drawing's quality too much (and hence the human's ability to read and interpret it), when the angles formed by the crossing edges are large. In other words, while prior to these experiments it was commonly accepted that mainly the number of crossings is the most important parameter for judging the quality of a non-planar graph drawing [14, 4], it turned out that the types of edge crossings also matter. As a result, a new and prominent research direction was initiated, recognized under the term "beyond planarity" [15, 16, 17], which focuses on graphs and their properties, when different constraints on the types of edges crossings are imposed; refer to [18] for a recent survey.

Formally, the value of the minimum angle formed by any two crossing edges in a drawing is referred to as its *crossing resolution*[1]. Analogously, the crossing resolution of a graph is defined as the maximum crossing resolution over all its drawings. Clearly, the crossing resolution of a non-planar graph is at most 90°, while a graph that admits a drawing with crossing resolution 90° is called *right-angle-crossing* (or *RAC*, for short) graph; see Figure 1 for an illustration. Notably, RAC graphs are sparse (an n-vertex RAC graph has at most $4n - 10$ edges [19]), while deciding whether a graph is RAC is NP-hard [20].

The latter result is an indication that the problem of finding drawings with high crossing resolution might also be difficult, even though, formally, its complexity has not been settled yet for values of the crossing resolution smaller than 90°. It is also worth noting that the literature is significantly more limited, when restricting the crossing resolution to be smaller than 90°, as also evidenced from our related work section.

[1] Also, refered to as *crossing angle resolution* in the literature.



(a) (b)

FIGURE 1: Illustration of: (a) a RAC drawing of the complete graph K_5, and (b) a drawing of the complete graph K_6, whose crossing resolution is arbitrarily close to 90°.

From a practical point of view, we are only aware of two methods that aim at drawings with high crossing resolution; both of them are adjustments of force-directed algorithms [7]. The first one is due to Argyriou et al. [21], while the second one is due to Huang et al. [22]. Common in both algorithms is that they apply appropriate forces on the endvertices of every pair of crossing edges. Each of them uses a different way to compute (the direction and the magnitude of) the forces, but the underlying idea of both is the same: the smaller the crossing angles are, the larger are the magnitudes of the forces applied at their endvertices.

In this work, we approach the crossing resolution maximization problem from a different perspective. We suggest a simple and intuitive randomization method for computing drawings with high crossing resolution, which, in a sense, mimics the way a human would try to increase the crossing resolution of a drawing. How would one increase the crossing resolution of a given drawing? First, she would try to identify the pair of edges that define the crossing resolution of the drawing (we call them *critical* edges); then, she would try to move an endvertex of this pair (which we choose at random), hoping that by this move the crossing resolution will increase. Of course, we cannot consider all possible positions for the vertex to be moved. Instead, we consider a small set of randomly generated ones. If there exists a position among them, that does not lead to a reduction of the crossing resolution, we move the vertex to this position.

Note that, in general, randomization is a technique that has not been deeply examined in Graph Drawing, as it seems difficult to even speculate about the expected quality of the produced drawings; a notable exception is the randomized approach by Goldschmidt and Takvorian [23] for computing large planar subgraphs of general graphs. Since we also could not provide any theoretical guarantee on the expected quality of the produced drawings (mainly due to the nature of the problem itself), we followed a more practical approach. We implemented our algorithm and the force-directed ones by Argyriou et al. [21] and by Huang et al. [22], and we experimentally compared them on standard benchmark graphs that are widely used in Graph Drawing for comparing different drawing algorithms. Our evaluation indicates that our method significantly outperforms the aforementioned force-directed algorithms in terms of crossing resolution, but this comes at the cost of slightly worse running time, especially for large and dense graphs. Analogous results are obtained, when our algorithm and the ones by Argyriou et al. [21] and by Huang et al. [22] are adjusted to maximize the *angular resolution* (that is, the minimum value of the angle between any two adjacent edges [24]) or the *total resolution* (that is, the minimum of the angular and the crossing resolution [21]).

Preliminaries: Unless otherwise specified, in this paper we consider simple undirected graphs. Let $G = (V, E)$ be such a graph. The degree of vertex $u \in V$ of G is denoted by $d(u)$. The degree $d(G)$ of graph G is defined as the maximum degree of its vertices, i.e., $d(G) = \max_{u \in V} d(u)$. Given a drawing $\Gamma(G)$ of G, we denote by $p(u) = (x_u, y_u)$ the position of vertex $u \in V$ of G in $\Gamma(G)$.

Related Work: As already mentioned, the study of the crossing resolution maximization problem has mainly focused on (properties of) RAC graphs, that is, on the optimal case of the crossing resolution maximization problem. Their study was initiated by Didimo et al. [19], who showed that an n-vertex RAC graph has at most $4n - 10$ edges [19], while deciding whether a graph is RAC is NP-hard [20]. The maximally-dense RAC graphs are 1-planar [25], i.e., they can be drawn with at most one crossing per edge. Actually, several relationships between the class of RAC graphs and subclasses of 1-planar graphs are known [26, 27]. Deciding, however, whether a 1-planar graph is RAC is NP-hard [28]. Note that the problem of finding RAC drawings has also been studied in the presence of bends [29, 30, 19, 31] and by imposing restrictions on the degree [32], the structure [33] and the drawing [34, 35] of the graph.

The results are fewer, when the right-angle constraint is relaxed. To the best of our knowledge, there is only one work, by Dujmovic et al. [36], which studies the crossing resolution maximization problem by relaxing the right-angle constraint on the crossing angles of the computed drawings. More precisely, in their work, Dujmovic et al. [36] proved that an n-vertex graph, whose crossing resolution is at least α (in radians) has at most $(3n - 6)\pi/\alpha$ edges. Corresponding density results for the case, in which few bends are allowed along each edge, are known by Ackerman et al. [37] and Di Giacomo et al. [31].

An immediate observation emerging from the above overview is that the focus has been primarily on theoretical aspects of the problem. Most of the approaches that could be useful in practice are based on

TABLE 1: Summary of the results for the Graph Drawing Contest 2017.

Graph	no. of vertices	no. of edges	CoffeeVM	TübingenMidnight	Time restricted	Our best
1	9	18	90°	77°	90°	90°
2	12	36	88°	42°	88°	89°
3	16	42	90°	89°	88°	90°
4	17	54	89°	89°	77°	89°
5	30	95	80°	30°	79°	87°
6	40	75	90°	78°	90°	90°
7	80	254	57°	34°	56°	64°
8	85	136	85°	61°	81°	89°
9	149	258	60°	9°	55°	88°
10	174	344	21°	4°	24°	24°
11	250	400	47°	6°	57°	72°
12	300	987	36°	5°	26°	36°
13	392	562	25°	4°	22°	34°
14	510	820	34°	5°	30°	43°
15	650	2,173	21°	4°	13°	29°
16	998	1,596	20°	3°	12°	26°

adjustments of force-directed techniques [7], according to which a graph is modelled as a physical system with forces acting on it, and a (good) drawing is obtained by an equilibrium of the system; for an introduction and a discussion of several variants we point the reader to [9].

More concretely, from an applicative point of view, Didimo et al. [38] describe a system, called *COWA*, to support conceptual web site traffic analysis [38]; its algorithmic core is a force-directed heuristic to compute simultaneous embeddings of two non-planar graphs with high crossing resolution. In a follow up work, Didimo et al. [39] describe heuristics, designed within the topology-driven force-directed framework, to achieve good trade-offs in terms of number of edge crossings, crossing resolution, and geodesic edge tendency. However, the obtained drawings are not straight-line. For straight-line drawings, Nguyen et al. [40] suggest a quadratic-programming based approach to increase the crossing angles of circular drawings. However, its usefulness is only limited to small graphs and to a specific kind of drawings.

Of more general scope are, of course, the already mentioned works by Argyriou et al. [21] and Huang et al. [22], which are based on the force-directed technique.

Motivation: Every year, the Graph Drawing conference organizes a contest, in which the task is to visualize a set of challenge graphs in a prescribed drawing style. The teams participating to the contest receive a set of challenge graphs, and after one hour they must submit their final drawings for each of them. The team with the highest cumulative score wins. In years 2017 and 2018, the task of the contest was to produce a drawing for each of the given challenge graphs with as high crossing resolution as possible (i.e., ignoring other aesthetic measures on the produced drawings). This task was actually our primary motivation for this work.

In 2017, we participated to the contest (by implementing a mixture of the two force-directed algorithms by Argyriou et al. [21] and by Huang et al. [22]). Table 1 summarizes the results, which are also available at `www.graphdrawing.de/contest2017/results.html`.

It is far from clear that the wining team was the team "`CoffeeVM`"; details for their approach can be found in [41]. Our team "`TuebingenMidnight`" miserably performed; we actually took the third position out of a total of four teams that participated. As already mentioned, each team had only one hour to compute a layout for each of the challenge graphs, which in year 2017 were in total 16. In Table 1, we additionally provide results that were achieved by the algorithm presented in this paper in two settings; within the time limit of the contest (see column "Time restricted"), as well as our best results, which were achieved without any time limitations (see column "Our best").

We can observe that for almost all graphs, our new approach achieves only slightly worse results than the ones of the contest's winner when restricted within the one hour. On a few graphs (namely, graphs 10 and 11), we even achieved better results. With a single exception (namely, graph 4), we could easily outperform the results that we achieved with the spring-embedder algorithm. If we neglect the time restriction, then for all the graphs, the results are (sometimes considerably) better than or at least about the same as the contest's winner. Nevertheless, what we realized is that the spring-embedder algorithm was not efficient enough for participating to the contest.

In 2018, we participated again to the Graph Drawing contest under the name "`TübingenColdShower`" (using an implementation of the algorithm presented in this paper). The winning team of the Graph Drawing contest 2017 also participated under the name "`CoffeeVM+`". Table 2 summarizes the results, which are also available at `www.graphdrawing.de/contest2018/results.html`.

TABLE 2: Summary of the results for the Graph Drawing Contest 2018.

Graph	no. of vertices	no. of edges	Grid size	TübingenColdShower	CoffeeVM+	Our best
1	10	32	100 × 100	87°	58°	88°
2	12	24	100 × 100	90°	90°	90°
3	18	49	150 × 150	88°	85°	90°
4	23	46	200 × 200	88°	60°	89°
5	30	91	200 × 200	71°	48°	78°
6	40	68	200 × 200	89°	71°	90°
7	50	94	250 × 250	80°	43°	83°
8	102	400	10,000 × 10,000	34°	27°	43°
9	128	256	10,000 × 10,000	77°	60°	83°
10	500	1,534	30,000 × 30,000	13°	19°	24°
11	709	1,602	50,000 × 50,000	11°	22°	18°
12	1,800	6,961	100,000 × 100,000	74°	9°	80°
13	1,500	4,467	1,000,000 × 1,000,000	2°	2°	4°
14	3,000	5,899	1,000,000 × 1,000,000	3°	11°	11°

Note that in year 2018, the organizers of the contest posed one additional constraint, namely, the drawing area was restricted to a certain size, which was different for each graph; see column "Grid size" in Table 2. It is eye-catching that our team was the winning one; team `CoffeeVM+` took the second position of the contest (out of a total of four teams that participated to the contest). It is worth noting that for almost all 14 challenge graphs our team achieved significantly better results than those of the second team `CoffeeVM+`, which clearly indicates that our new approach is very competitive (and definitely better than our previous, as also evidenced by the experimental evaluation).

Structure of the paper: The remainder of this paper is structured as follows. Our algorithm is presented in detail in Section 2 and is experimentally evaluated against the ones by Argyriou et al. [21] and by Huang et al. [22] in Section 3, where we also discuss our insights from this project. We conclude in Section 5 with a discussion of useful observations and insights that we obtained while working on this project.

2. DESCRIPTION OF OUR HEURISTIC APPROACH

In this section, we describe our heuristic for obtaining drawings with high crossing resolution. The input of our heuristic consists of a graph G and an initial drawing Γ_0 of G with crossing resolution $c(\Gamma_0)$. We assume that no two edges of graph G overlap in drawing Γ_0, i.e., $c(\Gamma_0) > 0$. A circular drawing or a drawing obtained by applying a force-directed algorithm on graph G clearly meets this precondition.

Our algorithm is iterative and at each iteration performs some operations that are mainly based on randomization. At the i-th iteration, we assume that we have computed a drawing Γ_{i-1} of graph G of crossing resolution $c(\Gamma_{i-1}) \geq c(\Gamma_0)$. In other words, we assume, as an invariant for our algorithm, that the crossing resolution cannot be decreased at some iteration of our algorithm (so, in a sense, our algorithm resembles probabilistic hill climbing approaches). We proceed by choosing a vertex of Γ_{i-1} arbitrarily at random based on the content of the so-called *vertex-pool*, which may contain:

- either a prespecified subset of the vertices of Γ_{i-1}, called *critical*, or
- all vertices of Γ_{i-1}.

Intuitively, the critical vertices are the endpoints of the edges that define the crossing resolution of drawing Γ_{i-1}. To formally define them, we first need to introduce the notion of critical edge-pairs. A pair of edges e and e' is called *critical* in drawing Γ_{i-1} if and only if edges e and e' cross in Γ_{i-1} and the minimum angle that is formed at their crossing point is equal to the crossing resolution of drawing Γ_{i-1} (that is, to $c(\Gamma_{i-1})$). The set of critical vertices of drawing Γ_{i-1} is then defined by the four endvertices of each critical edge-pair.

The role of critical vertices is central in our algorithm[2]: By appropriately changing the location of a critical vertex or of a vertex that is in the neighbourhood of the critical vertices, we naturally expect to improve the crossing resolution of the current drawing. We turned this observation into an algorithmic implementation through a probabilistic random selection procedure, so that the vertices at graph-distance i from the ones of the vertex-pool have higher probability for selection than the corresponding ones at graph-distance j in G, when $0 \leq i < j$. In other words, the closer a vertex is to the critical vertices, the more likely it is to be chosen. On the other hand, if the

[2]If the focus is not on the critical vertices for a large graph, then our algorithm will need a large number of iterations to converge to a solution with good crossing resolution, because it is simply very unlikely to select to move one of the vertices that define the crossing resolution.

vertex-pool contains all vertices of the graph, then each vertex can be chosen with the same probability.

What we quickly realized from our practical analysis, is that the crossing resolution of the initial drawing improves rapidly during the first iterations of the algorithm. However, by focusing only at the critical vertices, it is highly possible that the algorithm will get trapped to some local maxima after a number of iterations. Therefore, special care is needed to avoid these bottlenecks, especially when the input graph is large. We will discuss ways to avoid local maxima later in this section (refer to subsection 2.1).

So far, we have described the main idea of our algorithm, which at each iteration chooses uniformly at random a vertex of the current drawing to move (based on the content of the vertex-pool), so to improve the crossing resolution. Next, we describe how to compute the new position of the chosen vertex in the next drawing Γ_i. Recall that the crossing resolution $c(\Gamma_i)$ of drawing Γ_i that is obtained after the i-th iteration of the algorithm must be at least as large as the crossing resolution $c(\Gamma_{i-1})$ of drawing Γ_{i-1} (by the invariant of our algorithm), that is, $c(\Gamma_i) \geq c(\Gamma_{i-1})$.

Let v_i be the vertex of drawing Γ_{i-1} that has been chosen to be moved at the i-th iteration. To compute the position of vertex v_i in the next drawing Γ_i, we consider a set of ρ rays $r_0, r_1, \ldots, r_{\rho-1}$ that all emanate from $p(v_i)$ in Γ_{i-1}, such that the angle formed by ray r_j, with $j = 0, 1, \ldots, \rho - 1$, and the horizontal axis equals to $2j\pi/\rho$, where $\rho > 0$ is an integer parameter of the algorithm. These rays are then rotated by an angle that is chosen uniformly at random in the interval $[0, 2\pi]$; see Fig. 2 for an illustration. The position of vertex v_i in Γ_i will eventually be along one of the rays $r_0, r_1, \ldots, r_{\rho-1}$. More precisely, for each ray r_i we choose a distance value δ_i uniformly at random from the interval $[\delta_{min}, \delta_{max}]$, where δ_{min} and δ_{max} are two positive parameters of the algorithm. For each $j = 0, 1, \ldots, \rho-1$, a new point π_j is obtained by translating $p(u)$ along r_j by a distance δ_j; we call point π_j *feasible* if and only if there is no vertex of drawing Γ_{i-1} at point π_j and the crossing resolution of the drawing obtained by placing vertex v_i at point π_j and by keeping all other vertices of the graph in their positions in Γ_{i-1} is at least as large as the crossing resolution of Γ_{i-1}.

If none of the points π_j, with $j = 0, 1, \ldots, \rho - 1$ is feasible, then the position of v_i in Γ_i is $p(v_i)$, i.e., the same as in Γ_{i-1}, since $c(\Gamma_i) \geq c(\Gamma_{i-1})$ must hold. If there is one or more feasible points, then one may consider two different approaches to determine the position of vertex v_i in Γ_i. The most natural is to choose the feasible point that maximizes the crossing resolution of the obtained drawing. As an alternative, one may rely again on randomization and chose uniformly at random one of the feasible points as the position of vertex v_i in Γ_i. We note that we did not observe any significant difference between these two approaches (in terms of the crossing resolution of the obtained drawings), so we simply adopted the first one. The termination condition of our algorithm is simple and depends on an input parameter τ. More specifically, if the crossing resolution has not improved during the last τ iterations, we assume that the algorithm has converged and we stop.

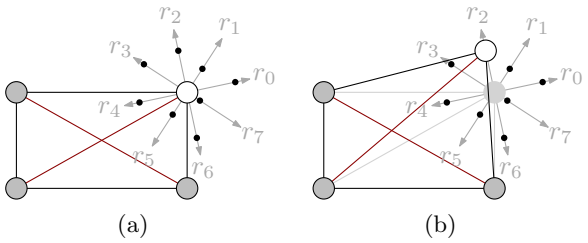


FIGURE 2: Illustration of an iteration step of our algorithm: (a) the chosen vertex is the white one; the computed rays $r_0, \ldots, r_7$ have been rotated by 8°; the black-colored points along these rays are points $\pi_0, \ldots, \pi_7$; among them, π_4 yields the best solution, and (b) the resulting drawing after moving the vertex at position π_2.

2.1. Avoiding local maxima

To avoid getting trapped to locally optimal solutions, we mainly investigated two approaches, which are both parametrizable by two input parameters ζ and ζ'. The first mimics the human behavior. What would one do to escape from a locally optimal solution? She would stop trying to move the endvertices of the edges defining the crossing resolution; she would rather start moving "irrelevant" vertices hoping that by doing so a better solution will be easier to be computed afterwards. Our algorithm is mimicking this idea as follows:

- if during the last ζ iterations the crossing resolution has not been improved, then the vertex-pool becomes *wider* by including all the vertices, and the algorithm is executed with this vertex-pool for ζ' iterations;
- afterwards, the vertex-pool switches back to the critical vertices.

From our extensive practical analysis, we quickly realized that, while this approach turned out to be effective for smaller graphs, for graphs with more than 100 vertices, it was not so efficient; in most iterations with the wider vertex-pool, the embedding could not change in a beneficial way for the algorithm to proceed.

Our second approach is based on parameters ρ, δ_{min} and δ_{max} of the algorithm. Our idea was that if the algorithm gets trapped to a locally optimal solution, then a "drastic" or "sharp" move may help to escape. We turned this idea into an algorithmic implementation as follows:

- if during the last ζ iterations the crossing resolution has not been improved, we double the values of ρ,

δ_{min} and δ_{max}, and the algorithm is executed with these values for ζ' iterations;

– afterwards, ρ, δ_{min} and δ_{max} switch back to this initial value.

Note, however, that this approach may lead to drawings with larger area, but this is "expected", as it turns out that drawings with high crossing resolution may require large area [29, 27].

2.2. Complexity issues

A factor that highly affects the efficiency of our algorithm is the computation of the crossing points of the edges and the corresponding angles at these points. Given a drawing, a naïve approach to compute its crossings requires $O(m^2)$ time, which can be improved by a plane-sweep technique to $O(m \log m + c)$ time, where m and c denote the number of edges and crossings; see, e.g., [42].

Instead of computing all crossing points and the corresponding angles for each candidate position of each iteration, we adopted a different approach for determining the set of feasible candidate positions, which turned out to be quite efficient in practice. Recall that we denoted by v_i the vertex chosen at the i-th iteration step, and by $\pi_0, \ldots, \pi_{\rho-1}$ the candidate positions to move v_i. Let $e_0, \ldots, e_{d_i-1}$ be the edges incident to v_i, where d_i denotes the degree of vertex v_i. Next, for each edge e_k with $k = 0, \ldots, d_i - 1$ we compute the crossings and the corresponding crossing angles of edge e_k with all other edges in Γ_{i-1}. Let ϕ_i be the minimum crossing angle computed; this is our reference angle. Also, for each candidate position π_j with $j = 0, \ldots, \rho - 1$, and for each edge e_k with $k = 0, \ldots, d_i - 1$, we compute the crossings and the corresponding crossing angles of e_k with all other edges of the drawing, assuming that vertex v_i is at position π_j. Let χ_j be the minimum crossing angle computed with this approach, when vertex v_i is at position π_j. Clearly, π_j is feasible, only if $\chi_j \geq \phi_i$. Note that the complexity of this approach is $O(\deg(v_i)m) = O(nm)$.

2.3. Some Interesting Variants

In general, aesthetically pleasant drawings of graphs are usually the result of compromising between different aesthetic criteria. Towards this direction, we discuss in this section interesting variants of our algorithm, which are motivated by the following observation that we made while working on this project (see Section 5): Drawings that are optimised only in terms of the crossing resolution tend to have bad aspect ratio (that is, the ratio of the longest to the shortest edge is large) and poor angular resolution (that is, the angles formed by adjacent edges are small). The former seems to be a consequence of the fact that drawings with good crossing resolution tend to be quite demanding in area. For the latter, observe that if in a drawing all edges are either almost horizontally or almost vertically drawn, such that only "horizontal" and "vertical" edges cross, then the crossing resolution of this drawing is arbitrarily close to 90°, while its angular resolution is arbitrarily close to 0°.

Aspect ratio. Formally, the aspect ratio of a drawing is the ratio of the length of its longest edge to the length of its shortest edge. Sometimes it is also used as a measure of the area of non-grid drawings. It was easy to instruct our algorithm to prevent producing drawings with aspect ratio either higher than the one of the starting layout or higher than a given input value. What we simply had to do was to reject candidate positions, which violate this precondition.

Total resolution. The notion of the total resolution of a drawing was introduced relatively recently with aim of "balancing" the measures of the crossing and of the angular resolution of a drawing [21]. Formally, it is defined as the minimum of these two measures. It was not difficult to adjust our algorithm to yield drawings with high total resolution by simply taking into account also the angular resolution of the drawing. In particular, if the total resolution of the drawing is defined by its angular resolution, then the way we compute the critical vertices of this drawing has to change; the critical vertices must be the endvertices of the pairs of edges that define the angular resolution. Also, at each iteration of our algorithm we have to ensure that the total resolution does not decrease. We do so by rejecting candidate positions which yield a reduced total resolution.

Angular resolution. As it is the case with the force-directed algorithms by Argyriou et al. [21] and by Huang et al. [22], our algorithm can also be restricted to maximize only the angular resolution (by neglecting its crossing resolution). We already described in the previous paragraph the necessary changes in the definition of the critical vertices and the rule according to which a candidate position is rejected (i.e., when it yields a drawing with a reduced angular resolution).

Grid drawings. Our algorithm, as it has been described so far, does not necessarily produce grid drawings, i.e., drawings in which the vertices are at integer coordinates. However, it can be easily adjusted to produce such drawings. More precisely, if we round the candidate positions computed at each iteration of our algorithm to their closest grid points and use these grid points as candidates for the next position of the vertex to be moved, then the obtained drawing will be grid (assuming, of course, that the starting drawing is grid). One can even bound the size of the grid, by rejecting candidate grid positions that lie outside the permitted bounds.

Note that an alternative approach is to let our algorithm compute a (non-grid) drawing and then apply

some technique to convert it to grid. However, critical in this step is not to affect the crossing resolution of the computed drawing too much during its conversion to grid. It is worth noting that we tried to apply such a technique (using again randomization). Our idea was to choose uniformly at random, for each vertex of the non-grid starting drawing, one of the grid points from its *neighbourhood*, and move it to this point (assuming of course that there is no other vertex at this point), where the neighbourhood of the vertex was initially set to its four closest grid points. If after a number of iterations a grid drawing without vertex-vertex overlaps and of similar crossing resolution as the one of the starting drawing could not be reported, then we had to augment the neighbourhood of each vertex so to contain more grid points (hoping that by this augmentation the computation of an acceptable grid drawing will be become feasible).

However, we faced two main difficulties when adopting this alternative approach. First, we could not always guarantee that the crossing resolution of the final grid drawing will be similar to the one of the starting drawing without increasing the drawing area too much. The second one is that the conversion itself is an extra step in the algorithm, which some times was more demanding than the main step of the algorithm, which was the actual computation of the drawing with high crossing resolution. So, we decided not to adopt this alternative approach.

3. EXPERIMENTAL EVALUATION

In this section, we present the results of our experimental evaluation. For comparison purposes, apart from our algorithm, we also implemented the force-directed algorithms by Argyriou et al. [21] and by Huang et al. [22]. The implementations[3] were in Java using the yFiles graph drawing library [43].

3.1. Experiment's Setup

As test sets for our experiment, we used the non-planar Rome graphs [44] and the non-planar AT&T graphs (also known as Graph Catalog or North graphs; available at `http://graphdrawing.org/data`), which are two collections of around 8.100 and 424 benchmark graphs, respectively, that are widely used in Graph Drawing for the evaluation of different drawing algorithms.

The experiment was performed on a Linux laptop with four cores at 2.4 GHz and 8 GB RAM. The experiment was performed as follows. Initially, each graph of our expertiment was laid out using the SmartOrganic layouter of yFiles [43]. Starting from this layout, every graph was drawn with:

- our algorithm with $\delta_{max} = \frac{1}{2}\max\{w, h\}$, where w and h are the width and the height of the initial drawing, respectively, $\delta_{min} = \frac{1}{100}\delta_{max}$ and $\rho = 10$,
- our algorithm restricted not to violate the aspect ratio of the initial layout,
- the force-directed algorithm by Argyriou et al., and
- the force-directed algorithm by Huang et al.

Since all algorithms of the experiment can easily be adjusted to maximize only the crossing resolution, or only the angular resolution or both (by maximizing the total resolution), we adjusted each of them to maximize exclusively the corresponding measures. In our algorithm, this can be achieved by modifying appropriately the content of the vertex-pool (as we discussed in Section 2.3), while in the algorithms of Argyriou et al. and of Huang et al. by switching on only the forces that maximize the corresponding properties under measure (note that, each of these two algorithms has a different set of forces to maximize the crossing and the angular resolution, such that together they maximize the total resolution). The reported results, illustrated in Figures 3–8, are on average across different drawings with same number of vertices.

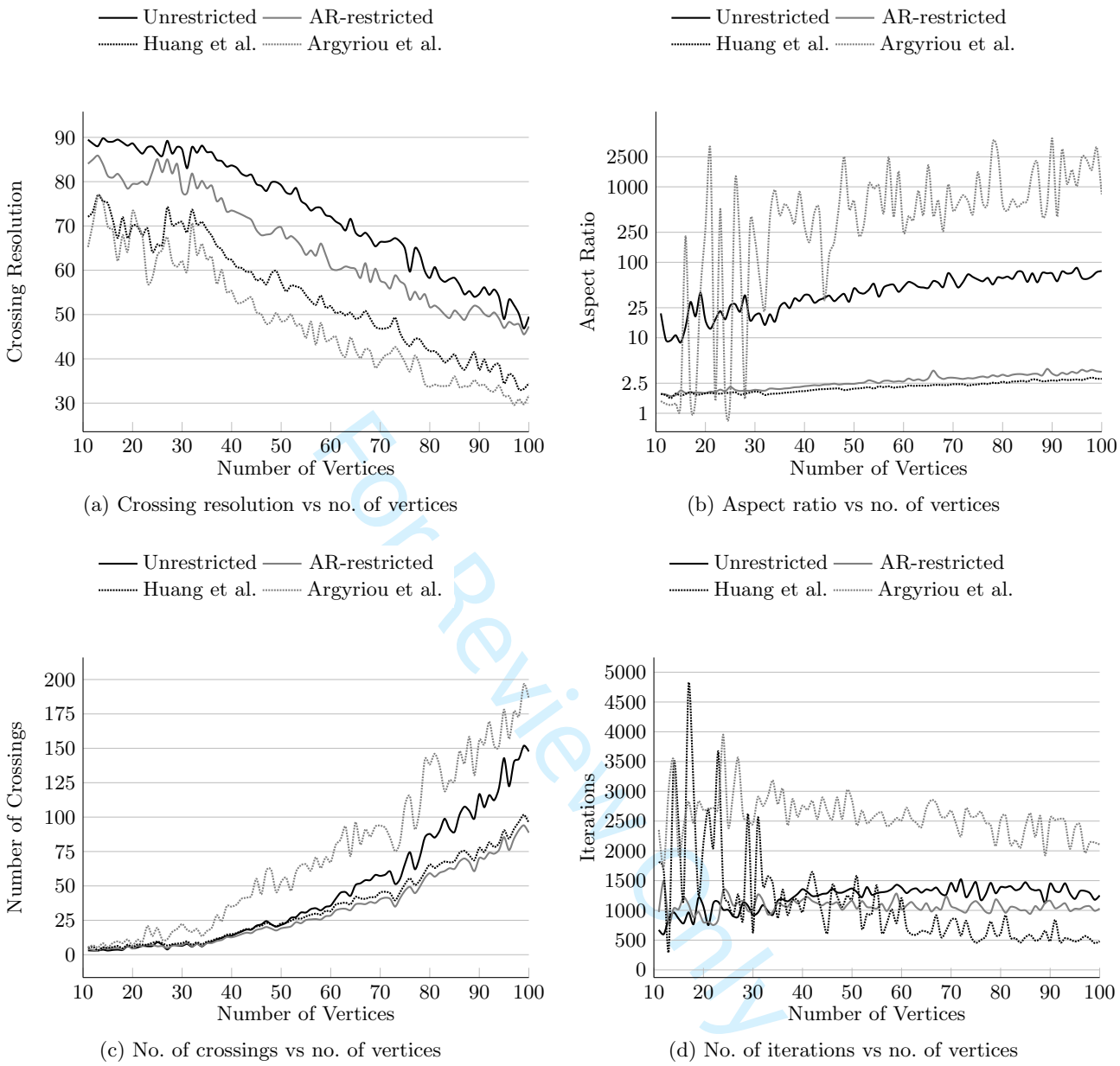
3.2. The Rome Graph Test Set

In this subsection, we report the results of our expertimental evaluation for the non-planar Rome graphs. Note that we did not impose any grid constraint on our algorithms. Our results are summarized in Figures 3, 5 and 4.

Crossing resolution. Our results for the crossing resolution are summarized in Fig. 3. Here, each algorithm was adjusted to maximize exclusively the crossing resolution (i.e., by ignoring the drawing's angular resolution). It is immediate to see that our algorithm outperforms all other ones in terms of the crossing resolution of the produced drawings, when we do not impose any restriction on the aspect ratio of the computed drawings; refer to the solid-black curve, denoted as *Unrestricted*, in Fig. 3a. The variant of our algorithm, which does not violate the aspect ratio of the initial layout, leads to drawings with slightly smaller crossing resolution; refer to the solid-gray curve, denoted as *AR-restricted*, in Fig. 3a. Finally, the two force-directed algorithms seem to produce drawings with worse crossing resolution; refer to the dotted-gray and dotted-black curves of Fig. 3a (by Argyriou et al. and by Huang et al., respectively).

While our unrestricted algorithm produces drawings with better crossing resolution, this comes at a cost of drastically increased aspect ratio (see Fig. 3b), which, however, is still better that the corresponding aspect ratio of the drawings produced by the algorithm of Argyriou et al. For the latter algorithm, it seems that the forces due to the angles formed at the crossings

[3] All implementations are available on request by the authors.



(a) Crossing resolution vs no. of vertices

(b) Aspect ratio vs no. of vertices

(c) No. of crossings vs no. of vertices

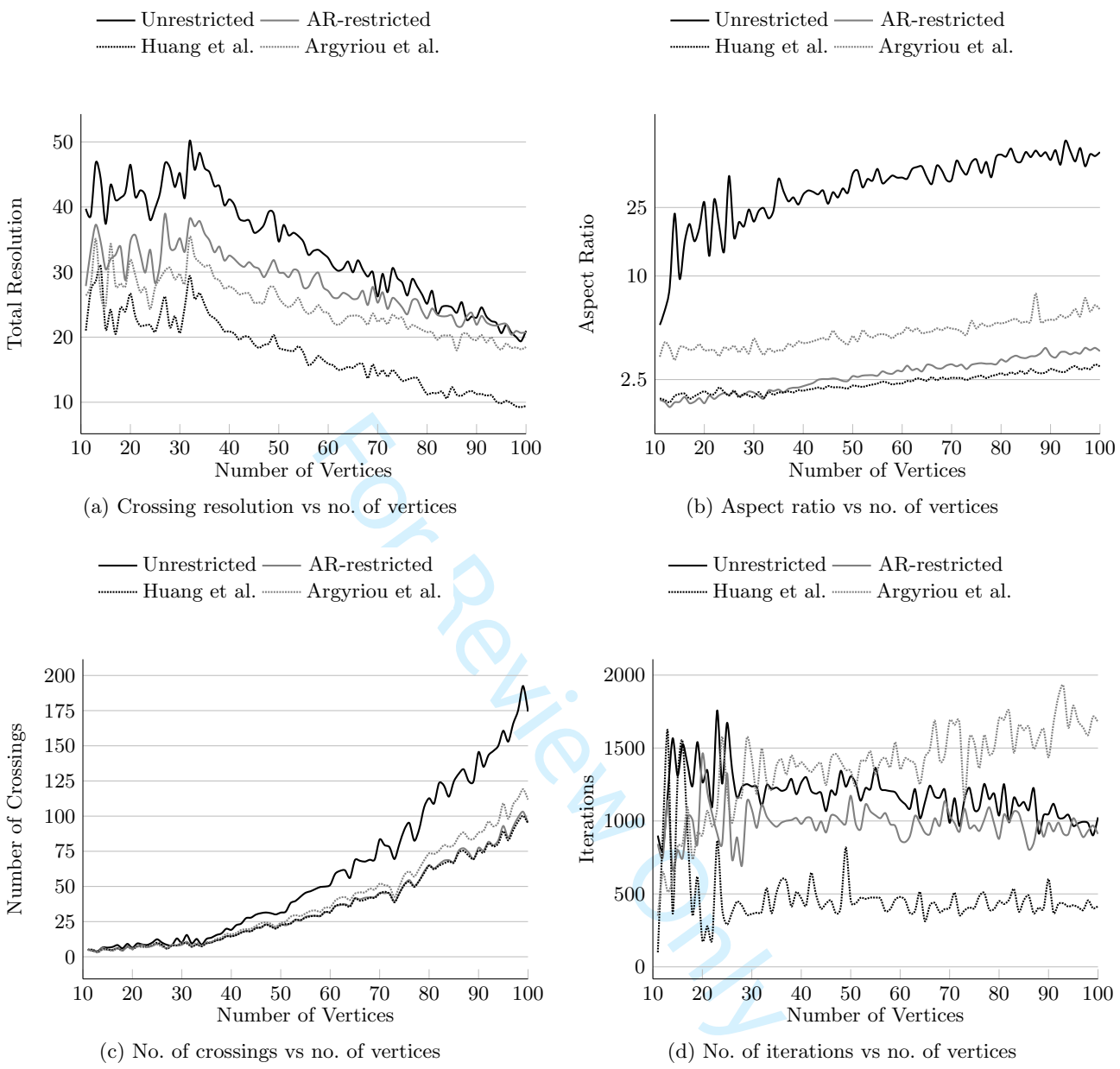
(d) No. of iterations vs no. of vertices

FIGURE 3: Experimental results on the crossing resolution for the Rome graphs.

outperform the corresponding spring forces, which try to keep the lengths of the edges short. Going back to our unrestricted algorithm, its behavior is up to a certain degree expected, mainly due to the fact that there is no control on the lengths of the edges (as it is, e.g., in the case of the force-directed algorithm of Huang et al. [22], whose sping forces inpose stong restrictions on the edge lengths). On the other hand, the restricted variant of our algorithm, which does not allow the aspect ratio to increase, has more or less comparable performance (in terms of aspect ratio) as the one of Huang et al.

Regarding the number of crossings, the restricted variant of our algorithm and the force-directed algorithm of Huang et al. yield drawings with comparable number of crossings, which at the same time is significantly smaller than the number of crossings produced by the two other algorithms; see Fig. 3c.

A different behavior can be observed in the number of iterations, which are required by the algorithms to converge; refer to Fig. 3d. We note here that we used different criteria to determine whether the algorithms of our experiment had converged. For our algorithms and for the force-directed algorithm by Huang et al., we assumed that the algorithm had converged, if the crossing resolution between 500 consecutive iterations was not improved by more than 0.001 degrees. For the algorithm by Argyriou et al., we decided to use a much more restricted convergence criterion, because the produced layouts can change vastly between consecutive iterations. We made this choice mainly



(a) Crossing resolution vs no. of vertices

(b) Aspect ratio vs no. of vertices

(c) No. of crossings vs no. of vertices

(d) No. of iterations vs no. of vertices

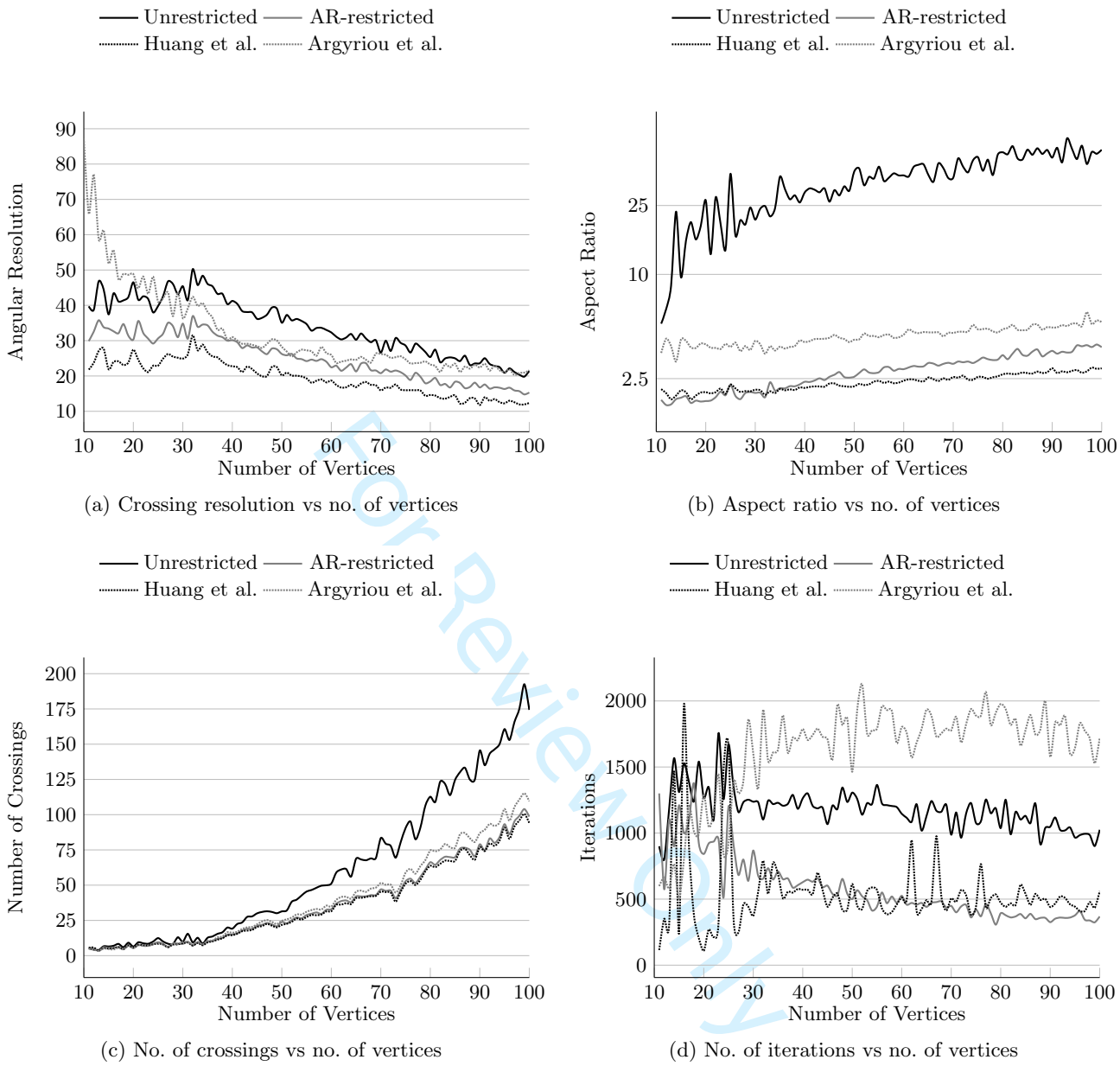
FIGURE 4: Experimental results on the total resolution for the Rome graphs.

to have "comparable" number of iterations among the algorithms of the experiment. In this direction, we adopted the convergence criterion that the authors used in their previous experimental analysis, that is, we assumed that the algorithm had converged, if the crossing resolution between two consecutive iterations was not improved by more than 0.001 degrees. Observe that even under this different convergence criterion, the algorithm needs significantly more iterations to converge than the remaining three algorithms of the experiment; see Fig. 3d.

The maximum number of iterations that each of the algorithms of the experiment could perform in order to converge was set to 100.000, but notably this limit was never reached.

We observe that both force-directed algorithms by Argyriou et al. and by Huang et al. seem to require a great amount of iterations to converge for small graphs, where a drawing with really good crossing resolution is more likely to be computed. However, that for larger graphs the algorithm by Huang et al. requires the least amount of iterations. Regarding the unrestricted and the restricted variants of our algorithm, it is clear that both require comparable number of iterations to converge, but clearly more than the ones of the algorithm by Huang et al.

Total resolution. Our results for the total resolution are summarized in Fig. 4. Here, each algorithm was adjusted to maximize both the crossing and the angular



(a) Crossing resolution vs no. of vertices

(b) Aspect ratio vs no. of vertices

(c) No. of crossings vs no. of vertices

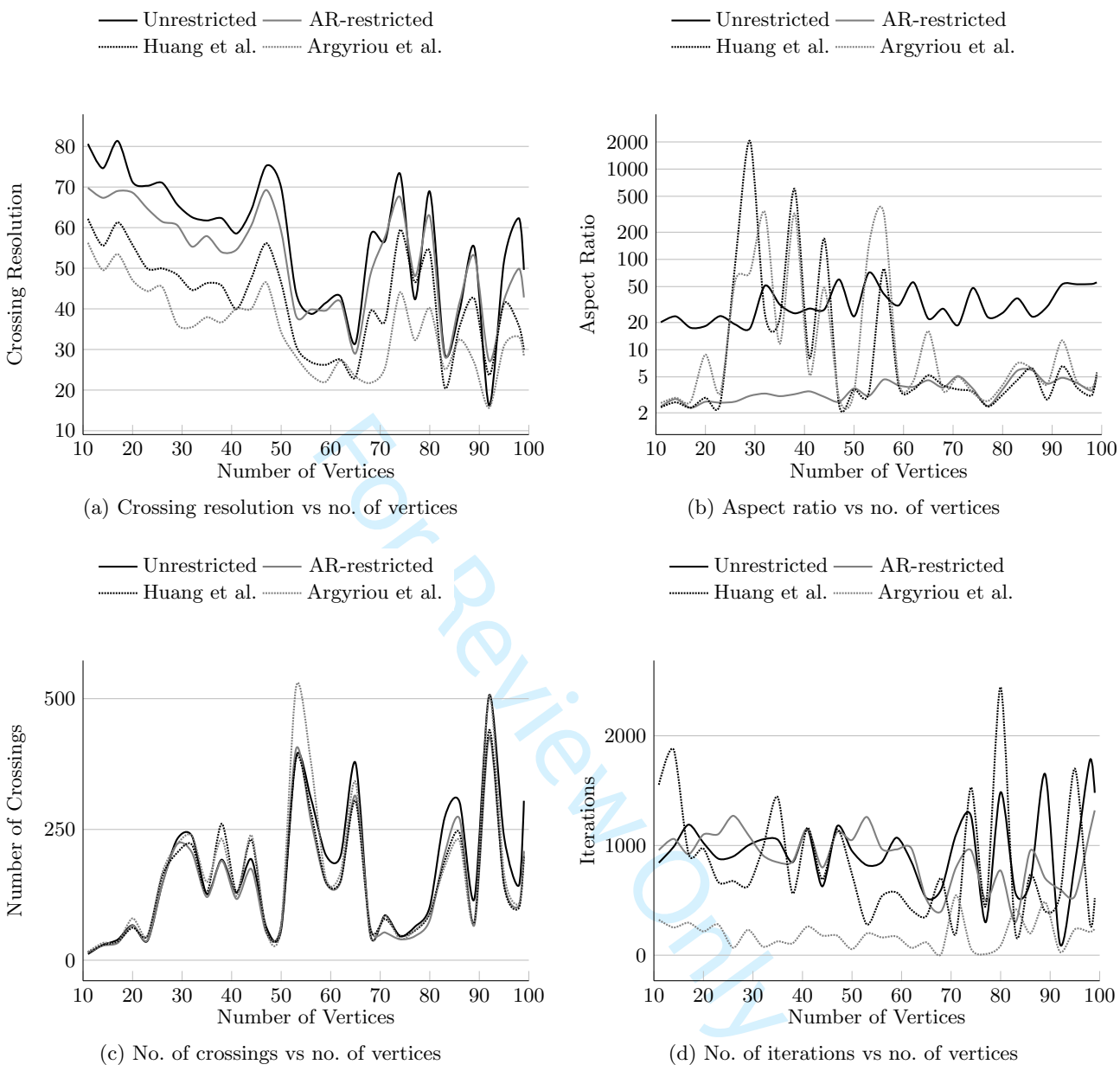
(d) No. of iterations vs no. of vertices

FIGURE 5: Experimental results on the angular resolution for the Rome graphs.

resolution, simultaneously. For the vast majority of the graphs in the experiment, both our unrestricted algorithm and its restricted variant yield drawings with better total resolution than the corresponding ones by Argyriou et al. The drawings produced by the algorithm by Huang et al. seems to have worse total resolution; see Fig. 4a. Note, however, that both variants of our algorithm as well as the force-directed algorithm by Argyriou et al. tend to produce drawings of the same total resolution for larger graphs with a small difference in our favor.

Contrary to the results for the total resolution, the results for the aspect ratio show that the drawings produced by the algorithm by Huang et al. are better (in terms of this measure) than the drawings produced by remaining algorithms; see Fig. 4b. The drawings produced by the restricted variant of our algorithm have slightly worse aspect ratios. Then, the ones produced by the force-directed algorithm by Argyriou et al. follow. Again, we observe that our unrestricted algorithm leads to drawings with very high aspect ratio.

Analogous observations can be made for the number of crossings of the produced layouts; see Fig. 4c. The restricted variant of our algorithm and the algorithm by Huang et al. yield drawings with the least number of crossings. Comparable but slightly worse (in terms of the number of crossings) are the drawings produced by the force-directed algorithm by Argyriou et al. Our unrestricted algorithm requires the largest number of crossings, which turns out to be notably higher than



(a) Crossing resolution vs no. of vertices

(b) Aspect ratio vs no. of vertices

(c) No. of crossings vs no. of vertices

(d) No. of iterations vs no. of vertices

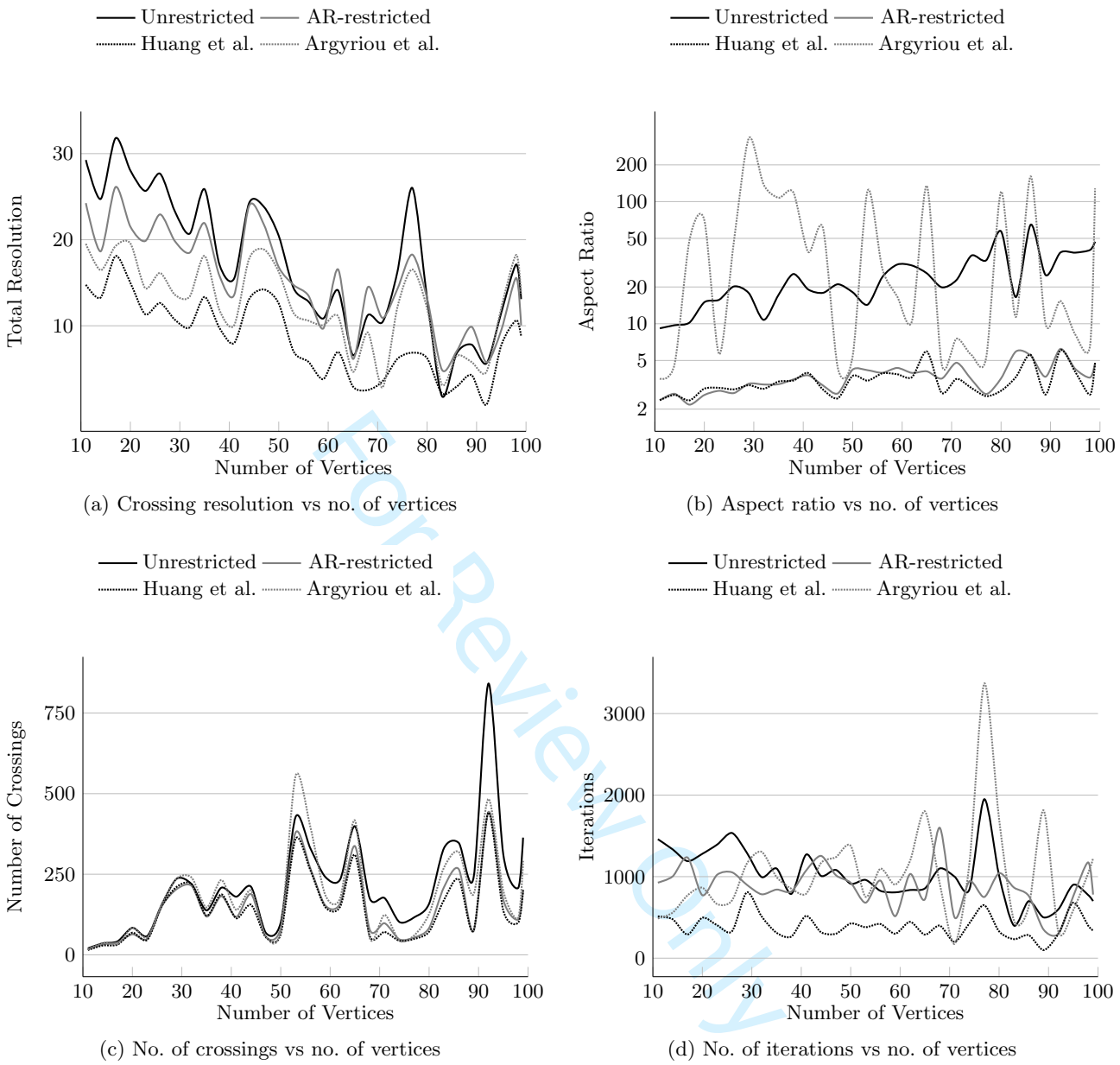
FIGURE 6: Experimental results for the crossing resolution on the AT&T graphs.

the corresponding ones of the other three algorithms.

On the negative side, both the unrestricted and the restricted variants of our algorithm require more iterations than the force-directed algorithm by Huang et al. in order to converge; see Fig. 4d. Recall, however, that the latter algorithm is clearly outperformed by both our variants in term of total resolution. The algorithm by Argyriou et al. clearly requires the highest number of iterations (especially for large graphs). We note that the convergence criterion was the same as for the crossing resolution; however, the measured quality was (not the crossing but) the total resolution.

Angular resolution. We conclude the analysis of our experimental evaluation with the results for the angular resolution; see Fig. 5. Here, each algorithm was adjusted to maximize only the angular resolution (i.e., by ignoring the drawing's crossing resolution). A notable observation is that, for small graphs the best results are achieved by the algorithm by Argyriou et al., while for medium-size graphs by our unrestricted algorithm; see Fig. 5a. For large graphs, the two algorithms tend to have the same performance. The restricted variant of our algorithm yields drawings with slightly worse angular resolution. The algorithm by Huang et al. is outperformed by all algorithms of the experiment.

The results for the aspect ratio, the number of crossings and the required number of iterations are very similar with corresponding ones for the total resolution;



(a) Crossing resolution vs no. of vertices

(b) Aspect ratio vs no. of vertices

(c) No. of crossings vs no. of vertices

(d) No. of iterations vs no. of vertices

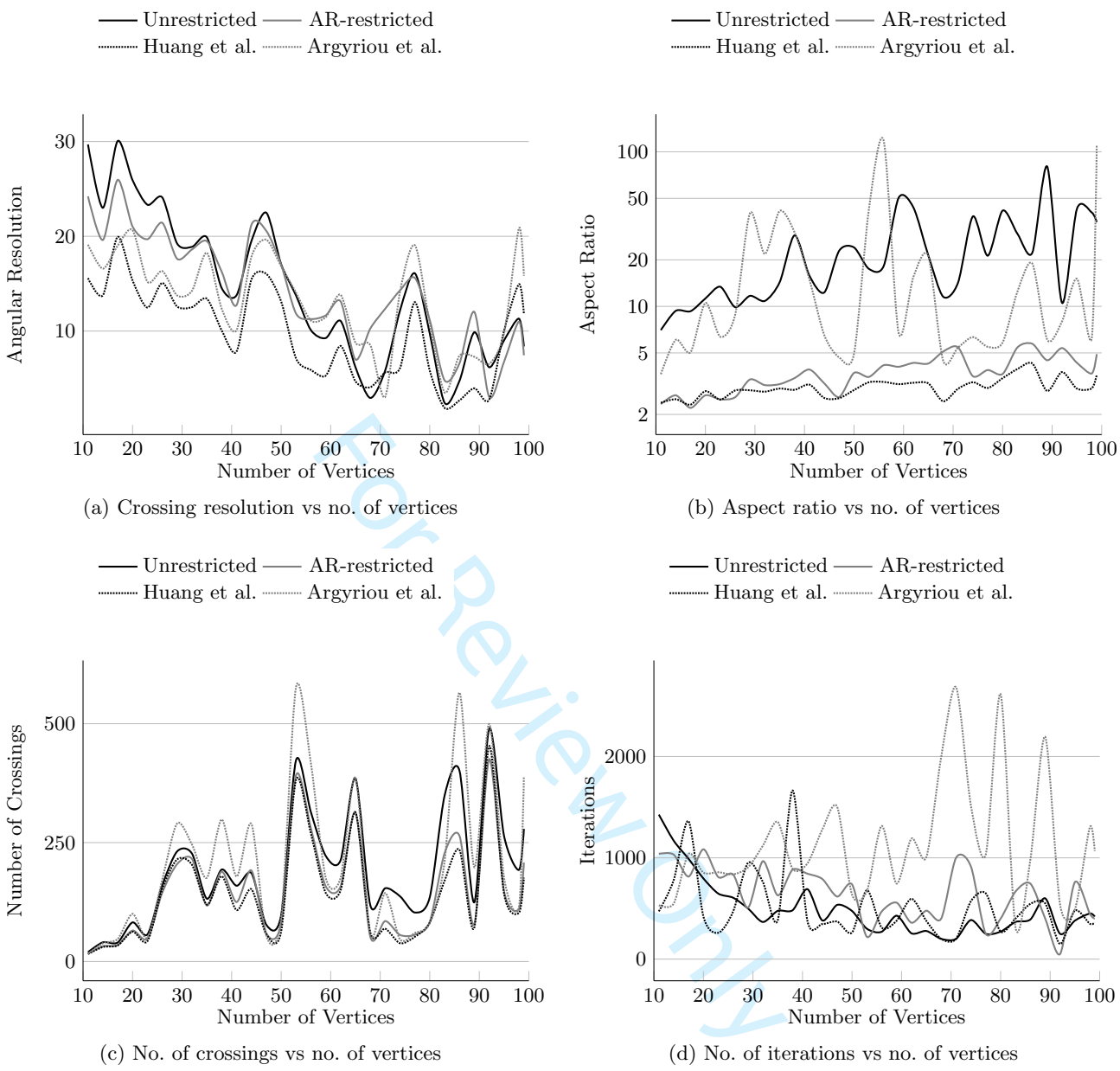
FIGURE 7: Experimental results for the total resolution on the AT&T graphs.

see Figs. 5b–5d. This observation suggests that, for most of the graphs of our experiment, the angular resolution dominates the crossing resolution (and thus is the one defining the total resolution) in the constructed drawings, which explains the similarity in the reported results. The small differences result from the fact that the crossing resolution cannot be entirely neglected.

3.3. The AT&T Graph Test Set

In this subsection, we report the results of our experimental evaluation for the non-planar AT&T graphs. Note that again we did not impose any grid constraint on our algorithms. The corresponding results are illustrated in Figs. 6, 7 and 8. In general, we observed that the variance of the results is much larger than in the experiments on the Rome graphs. This manifests in spikes of large magnitude in the illustrations of the results indicates that the structural properties of the graphs in this second test set varies vastly between graphs of different sizes.

Crossing resolution. As with the Rome graphs, we observed again that both variants of our algorithm again outperformed the two force-directed algorithms; see Fig. 6a. Remarkable is the synchronous behavior of all four algorithms regarding the crossing resolution, as the curves are nearly parallel. The results actually indicate that one can classify these graphs into “hard” and “easy” ones, when maximizing their crossing



(a) Crossing resolution vs no. of vertices

(b) Aspect ratio vs no. of vertices

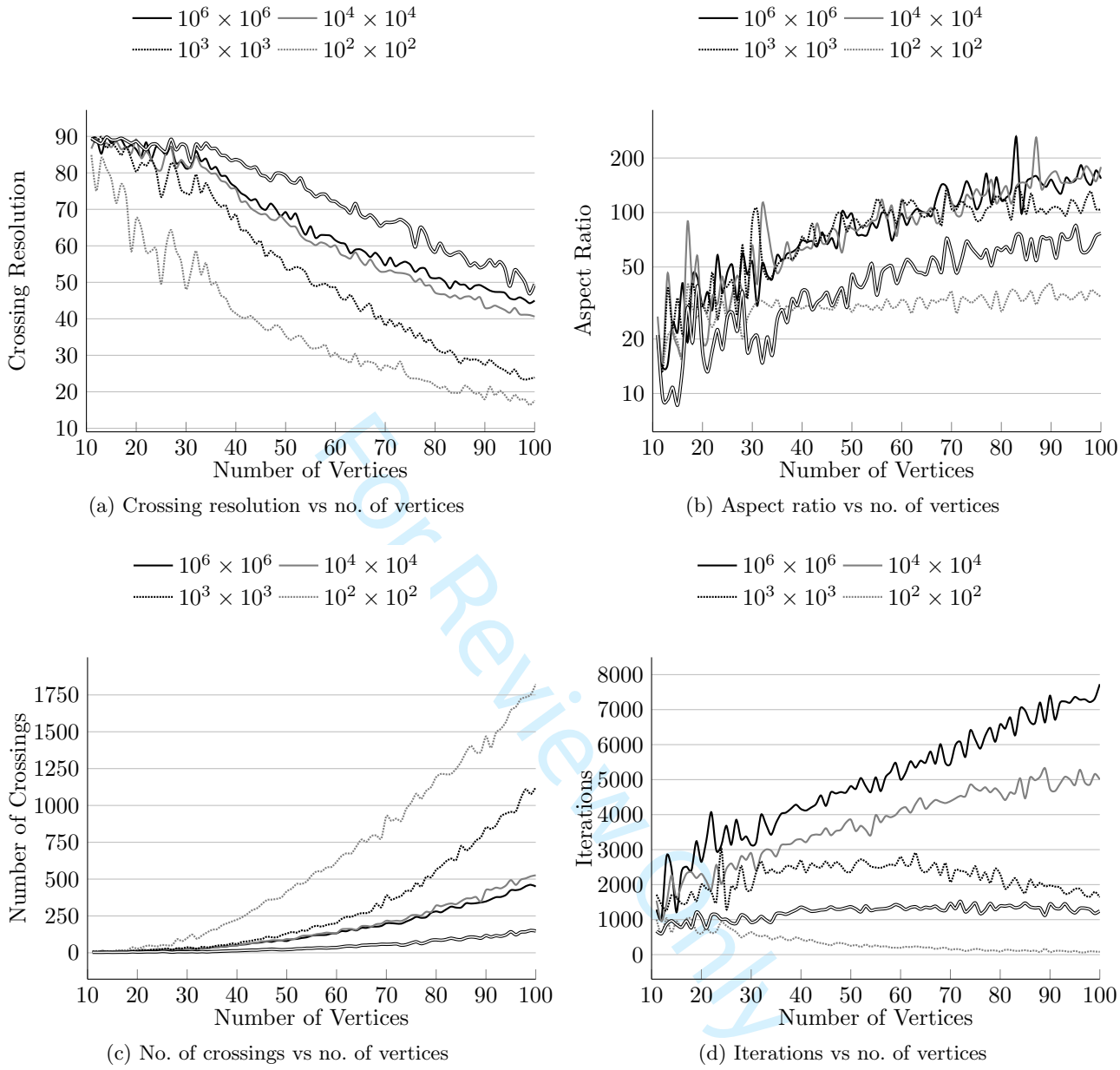
(c) No. of crossings vs no. of vertices

(d) No. of iterations vs no. of vertices

FIGURE 8: Experimental results for the angular resolution on the AT&T graphs.

resolution. In particular, the graphs with 50 to 70 vertices appear to be harder than the graphs with 70 to 80 vertices. As expected, the aspect ratio of the drawings produced by both the unrestricted and the restricted variants of our algorithm depends vastly on the number of vertices of the graphs. However, this natural behavior cannot be observed in the resulting drawings of the two force-directed algorithms, which appear to be quite unstable resulting in a large variance. Again the restricted variant of our algorithm and the two force-directed algorithms produce drawings with similar aspect ratio, which is much lower than the one of our unrestricted algorithm for larger graphs. All four algorithms behave nearly the same in terms of the number of crossings; see Fig. 6c. In terms of the number of iterations need for convergence, we observe that somewhat surprisingly the algorithm of Argyriou et al. converges using the least amount of iterations, while the remaining three algorithms behave nearly the same; see Fig. 6d.

Total resolution. For small graphs, we observed similar results as in the experiment on the Rome graphs, that is, our unrestricted algorithm outperforms (in terms of the total resolution) the other three algorithms of the experiment, while the restricted variant of our algorithm is slightly better than the algorithms by Argyriou et al. and by Huang et al.; see Fig. 7a. For larger graphs, however, the two variants of our algorithm achieve similar results, while still



(a) Crossing resolution vs no. of vertices

(b) Aspect ratio vs no. of vertices

(c) No. of crossings vs no. of vertices

(d) Iterations vs no. of vertices

FIGURE 9: Our experimental results on the crossing resolution with different grid restrictions. The double line corresponds to our unrestricted algorithm.

outperforming the algorithm by Huang et al. Regarding the aspect ratio and the number of crossings of the produced drawings, we observe that all algorithms have more or less the same behavior as when adjusted for maximizing the crossing resolution only. The only exception can be observed in the algorithm by Huang et al., which performs more stable; see Figs. 7b and 7c. With respect to the number of iterations required for convergence, the two variants of our algorithms and the force-directed algorithm by Argyriou et al. show similar behavior. The algorithm by Huang et al., on the other hand, needs less iterations for convergence; see Fig. 7d. Note, however, that there is no immediate correlation with the number of vertices of the graph.

Angular resolution. Here, the picture is not so clear concerning the ranking of the algorithms; see Fig. 8a. Only the algorithm by Huang et al. seems to be mostly at last in the rank. Concerning the aspect ratio, we see very good behavior for the restricted variant of our algorithm and the algorithm by Huang et al., while the remaining two algorithms show large variance and much worse values; see Fig. 8b. Regarding the number of crossings, we again observe that all algorithms have similar behavior. Note, however, that the unrestricted variant of our algorithm and the algorithm by Argyriou et al. yield drawings with more crossing, especially for graphs with large number of vertices; see Fig. 8c. Finally, both variants of our algorithm and the force-

directed algorithm by Huang et al. need comparable number of iterations for convergence, which most of the times is lower than the corresponding one by Argyriou et al.; see Fig. 8d.

Summarizing, we conclude that compared to the Rome graphs, the AT&T graphs show a much higher variance regarding the various resolution measures.

3.4. Grid Restricted Drawings

In addition to the experiments that we have described so far, we also evaluated how our algorithm performs, if we restrict its vertices to integer grid coordinates. In particular, we were interesting to see how the different quality measures that we have investigated so far are affected by the restrictions imposed on having the vertices of the graph on integer grids of different sizes:

(i) $10^6 \times 10^6$

(ii) $10^4 \times 10^4$

(iii) $10^3 \times 10^3$, and

(iv) $10^2 \times 10^2$.

The test suite for this experiment was the non-planar Rome graphs. The algorithm to be evaluated was the variant of our algorithm that optimizes the drawing's crossing resolution (without any restriction on the aspect ratio). However, since this algorithm is guaranteed to produce a grid drawing, only if its initial drawing is grid, each of the Rome graphs was initially laid out by randomly placing its vertices on the underlying grid, ensuring that neither two vertices nor two edges overlap. Fig. 9 summarizes the results.

Regarding the crossing resolution, we natural observe that the bigger the grid is, the easier it is to compute a drawing with high crossing resolution; see Fig. 9a. More precisely, the grid of size $10^2 \times 10^2$ was too restrictive for the vast majority of the graphs. As a result, the reported drawings were often the initial ones (as our algorithm could not improve them), especially for large graphs. Significantly fewer were the graphs for which our algorithm could not report an improved drawing, when the underlying grid was of size $10^3 \times 10^3$. The crossing resolution of the drawings produced by our algorithm, when the underlying grid was of size $10^4 \times 10^4$, were on average worse by only 10° than those produced by our algorithm, when imposing no restriction on the size of the underlying grid (refer to the double line in Fig. 9a). As expected, the gap was more narrow, when the underlying grid was of size $10^6 \times 10^6$.

The aspect ratio of the computed drawings was more or less the same regardless of the size of the underlying grid, with the exception of the drawings computed on the $10^2 \times 10^2$ grid; see Fig. 9b. The fact that the aspect ratio of these drawings was worse can be explained of course by the fact that in most cases an improved drawing could not be reported.

As expected, the smaller the underlying grid is, the more crossings the computed drawings contain; see Fig. 9c. As a result, the drawings produced by our algorithm, when imposing no restriction on the size of the underlying grid, are clearly better (in terms of the number of crossings) than those produced by our algorithm, when the size of the underlying grid is limited. As a matter of fact, the larger the underlying grid is, the fewer crossing the reported drawings have, and the differences are quite clear. Exceptional are the grids of sizes $10^4 \times 10^4$ and $10^6 \times 10^6$, where the improvement (in the number of crossings) from the latter to the former is small on average; this can partially be explained by the different choices of the initial drawings.

In general, the number of iterations needed for convergence increases with the size of the underlying grid; see Fig. 9d. Exceptional seems to be the $10^2 \times 10^2$ grid, which verifies our previous observation that for the vast majority of the graphs an improved drawing could not be reported.

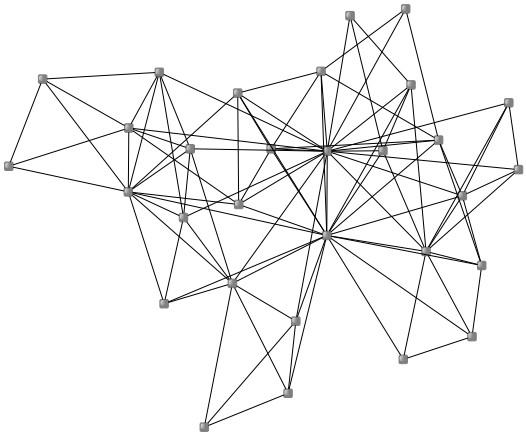
In conclusion, we can state that our algorithm is still able to compute drawings with high crossing resolution, when restricted to a grid, as long as the grid is not too small. However, the computation of a grid drawing takes longer, which is of course expected. Finally, the choice of the initial grid drawing seems to affect the performance of our algorithm, both with respect to the quality of the produced drawings (counted here in terms of the crossing resolution), but also with respect to the number of iterations needed for convergence.

4. SAMPLE DRAWINGS

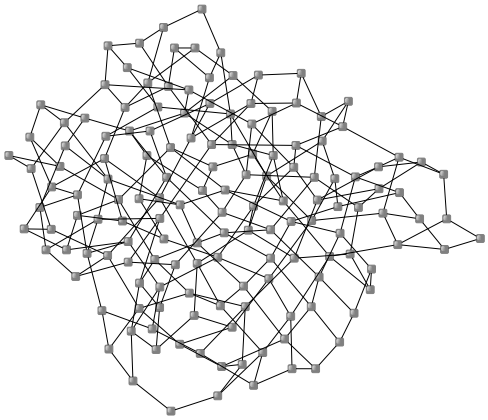
In this section, we present sample drawings of the 5th and of the 9th graph given in the Graph Drawing contest 2017 that are produced by different variants of our algorithm and of the algorithms by Argyriou et al. [21], and by Huang et al. [22]; see Figures 11 and 12, respectively. Each variant was obtained by adjusting each of the aforementioned algorithms to optimize the crossing resolution, the angular resolution and the total resolution, respectively; the aesthetic criterion optimized by each variant is reported in the caption of its corresponding subfigure. As initial drawings for all algorithms, we used the layouts shown in Fig. 10 that we computed with the SmartOrganic layouter of the yFiles library [43].

5. CONCLUSIONS

In this paper, we introduced a new heuristic aiming to produce drawings of high crossing resolution, for which we also presented variants that take into account other common aesthetic criteria in Graph Drawing. Our experimental evaluation indicates that the new heuristic is competitive to the state of the art force-directed algorithms (even when restricted to a given maximum



(a)



(b)

FIGURE 10: Illustration of (a) the 5th, and of (b) the 9th graph of the Graph Drawing contest 2017.

aspect ratio). For the latter algorithms, our evaluation indicates that their performance is good, only when several aesthetic criteria are taken into account.

It also worth noting that while working on this project, we made some useful observations and obtained some interesting insights. In particular, there is a recent observation (also supported by experiments) that drawings, in which the crossing angles are large, are easy to read and understand. We observed that drawings that are optimized only in terms of the crossing angles might be arbitrarily bad and may have several undesired properties. In particular, in these drawings it was very common to have adjacent edges to run almost in parallel and vertices to be very close to each other (see, e.g., Fig. 11g). Hence, their angular resolution and aspect ratio were often poor. The additional restrictions that we imposed regarding the angular resolution and the aspect ratio helped in significantly improving the readability of the produced drawings, without loosing too much of their quality in terms of the crossing resolution.

The framework that we developed seems to be quite adaptable to optimize or to take into account also other desired aesthetic properties of a drawing. Thus, as a future direction, we plan to further extend and evaluate our technique, e.g., by using a multi-criteria objective function (i.e., that is not limited to the crossing angles) that takes into account more aesthetic criteria.

Acknowledgments. The authors would like to thank Simon Wegendt and Jessica Wolz for implementing the first version of our prototype. This work is supported by the DFG grant Ka812/17-1. A preliminary version of this work has appeared in the Proceedings of the 26th International Symposium on Graph Drawing and Network Visualization (GD 2018).

REFERENCES

[1] de Fraysseix, H., Pach, J., and Pollack, R. (1990) How to draw a planar graph on a grid. *Combinatorica*, **10**, 41–51.

[2] Gutwenger, C. and Mutzel, P. (1998) Planar polyline drawings with good angular resolution. In Whitesides, S. (ed.), *Graph Drawing*, LNCS, **1547**, pp. 167–182. Springer.

[3] Kant, G. (1996) Drawing planar graphs using the canonical ordering. *Algorithmica*, **16**, 4–32.

[4] Purchase, H. C. (2000) Effective information visualisation: a study of graph drawing aesthetics and algorithms. *Interacting with Computers*, **13**, 147–162.

[5] Eades, P. and Wormald, N. C. (1994) Edge crossings in drawings of bipartite graphs. *Algorithmica*, **11**, 379–403.

[6] Sugiyama, K., Tagawa, S., and Toda, M. (1981) Methods for visual understanding of hierarchical system structures. *IEEE Trans. Systems, Man, and Cybernetics*, **11**, 109–125.

[7] Eades, P. (1984) A heuristic for graph drawing. *Congressus Numerantium*, **42**, 149–160.

[8] Fruchterman, T. M. J. and Reingold, E. M. (1991) Graph drawing by force-directed placement. *Softw., Pract. Exper.*, **21**, 1129–1164.

[9] Di Battista, G., Eades, P., Tamassia, R., and Tollis, I. G. (1999) *Graph Drawing: Algorithms for the Visualization of Graphs*. Prentice-Hall.

[10] Kaufmann, M. and Wagner, D. (eds.) (2001) *Drawing Graphs, Methods and Models*, LNCS, **2025**. Springer.

[11] Tamassia, R. (ed.) (2013) *Handbook on Graph Drawing and Visualization*. Chapman and Hall/CRC.

[12] Huang, W. (2007) Using eye tracking to investigate graph layout effects. In Hong, S. and Ma, K. (eds.), *APVIS*, pp. 97–100. IEEE Computer Society.

[13] Huang, W., Eades, P., and Hong, S. (2014) Larger crossing angles make graphs easier to read. *J. Vis. Lang. Comput.*, **25**, 452–465.

[14] Purchase, H. C. (1997) Which aesthetic has the greatest effect on human understanding? In Battista, G. D. (ed.), *Graph Drawing*, Lecture Notes in Computer Science, **1353**, pp. 248–261. Springer.

[15] Hong, S. and Tokuyama, T. (2016). Algorithmics for beyond planar graphs. NII Shonan Meeting Seminar 089.

[16] Kaufmann, M., Kobourov, S., Pach, J., and Hong, S. (2016). Beyond planar graphs: Algorithmics and combinatorics. Dagstuhl Seminar 16452.

[17] Liotta, G. (2017). Graph drawing beyond planarity: Some results and open problems. SoCG Week, Invited talk.

[18] Didimo, W., Liotta, G., and Montecchiani, F. (2018) A survey on graph drawing beyond planarity. *CoRR*, **abs/1804.07257**.

[19] Didimo, W., Eades, P., and Liotta, G. (2011) Drawing graphs with right angle crossings. *Theor. Comput. Sci.*, **412**, 5156–5166.

[20] Argyriou, E. N., Bekos, M. A., and Symvonis, A. (2012) The straight-line RAC drawing problem is NP-hard. *J. Graph Algorithms Appl.*, **16**, 569–597.

[21] Argyriou, E. N., Bekos, M. A., and Symvonis, A. (2013) Maximizing the total resolution of graphs. *Comput. J.*, **56**, 887–900.

[22] Huang, W., Eades, P., Hong, S., and Lin, C. (2013) Improving multiple aesthetics produces better graph drawings. *J. Vis. Lang. Comput.*, **24**, 262–272.

[23] Goldschmidt, O. and Takvorian, A. (1994) An efficient graph planarization two-phase heuristic. *Networks*, **24**, 69–73.

[24] Formann, M., Hagerup, T., Haralambides, J., Kaufmann, M., Leighton, F. T., Symvonis, A., Welzl, E., and Woeginger, G. J. (1993) Drawing graphs in the plane with high resolution. *SIAM J. Comput.*, **22**, 1035–1052.

[25] Eades, P. and Liotta, G. (2013) Right angle crossing graphs and 1-planarity. *Discrete Applied Mathematics*, **161**, 961–969.

[26] Bachmaier, C., Brandenburg, F. J., Hanauer, K., Neuwirth, D., and Reislhuber, J. (2017) Nic-planar graphs. *Discrete Applied Mathematics*, **232**, 23–40.

[27] Brandenburg, F. J., Didimo, W., Evans, W. S., Kindermann, P., Liotta, G., and Montecchiani, F. (2016) Recognizing and drawing ic-planar graphs. *Theor. Comput. Sci.*, **636**, 1–16.

[28] Bekos, M. A., Didimo, W., Liotta, G., Mehrabi, S., and Montecchiani, F. (2017) On RAC drawings of 1-planar graphs. *Theor. Comput. Sci.*, **689**, 48–57.

[29] Angelini, P., Cittadini, L., Didimo, W., Frati, F., Di Battista, G., Kaufmann, M., and Symvonis, A. (2011) On the perspectives opened by right angle crossing drawings. *J. Graph Algorithms Appl.*, **15**, 53–78.

[30] Arikushi, K., Fulek, R., Keszegh, B., Moric, F., and Tóth, C. D. (2012) Graphs that admit right angle crossing drawings. *Comput. Geom.*, **45**, 169–177.

[31] Giacomo, E. D., Didimo, W., Liotta, G., and Meijer, H. (2011) Area, curve complexity, and crossing resolution of non-planar graph drawings. *Theory Comput. Syst.*, **49**, 565–575.

[32] Angelini, P., Di Battista, G., Didimo, W., Frati, F., Hong, S., Kaufmann, M., Liotta, G., and Lubiw, A. (2011) Large angle crossing drawings of planar graphs in subquadratic area. In Márquez, A., Ramos, P., and Urrutia, J. (eds.), *EGC*, LNCS, **7579**, pp. 200–209. Springer.

[33] Didimo, W., Eades, P., and Liotta, G. (2010) A characterization of complete bipartite RAC graphs. *Inf. Process. Lett.*, **110**, 687–691.

[34] Giacomo, E. D., Didimo, W., Eades, P., and Liotta, G. (2014) 2-layer right angle crossing drawings. *Algorithmica*, **68**, 954–997.

[35] Hong, S. and Nagamochi, H. (2015) Testing full outer-2-planarity in linear time. In Mayr, E. W. (ed.), *WG*, LNCS, **9224**, pp. 406–421. Springer.

[36] Dujmovic, V., Gudmundsson, J., Morin, P., and Wolle, T. (2011) Notes on large angle crossing graphs. *Chicago J. Theor. Comput. Sci.*, **2011**.

[37] Ackerman, E., Fulek, R., and Tóth, C. D. (2012) Graphs that admit polyline drawings with few crossing angles. *SIAM J. Discrete Math.*, **26**, 305–320.

[38] Didimo, W., Liotta, G., and Romeo, S. A. (2010) Graph visualization techniques for conceptual web site traffic analysis. *IEEE PacificVis*, pp. 193–200. IEEE Computer Society.

[39] Didimo, W., Liotta, G., and Romeo, S. A. (2010) Topology-driven force-directed algorithms. In Brandes, U. and Cornelsen, S. (eds.), *Graph Drawing*, LNCS, **6502**, pp. 165–176. Springer.

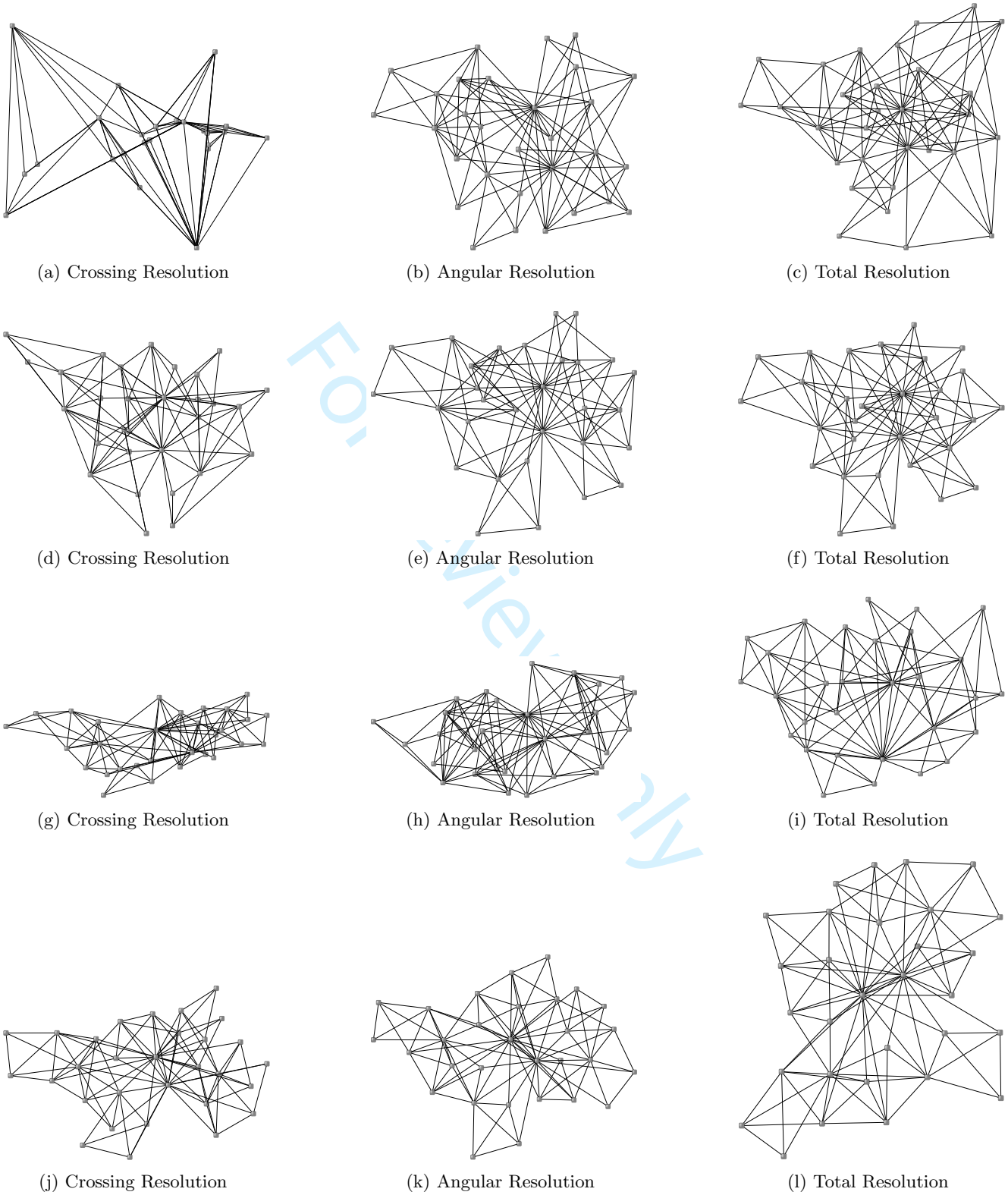
[40] Nguyen, Q. H., Eades, P., Hong, S., and Huang, W. (2010) Large crossing angles in circular layouts. In Brandes, U. and Cornelsen, S. (eds.), *Graph Drawing*, LNCS, **6502**, pp. 397–399. Springer.

[41] Demel, A., Dürrschnabel, D., Mchedlidze, T., Radermacher, M., and Wulf, L. (2018) A greedy heuristic for crossing-angle maximization. *CoRR*, **abs/1807.09483**.

[42] de Berg, M., Cheong, O., van Kreveld, M. J., and Overmars, M. H. (2008) *Computational geometry: algorithms and applications, 3rd Edition*. Springer.

[43] Wiese, R., Eiglsperger, M., and Kaufmann, M. (2004) *yFiles* - visualization and automatic layout of graphs. *Graph Drawing Software*, pp. 173–191. Springer.

[44] Di Battista, G. and Didimo, W. (2013) Gdtoolkit. In Tamassia, R. (ed.), *Handbook on Graph Drawing and Visualization.*, pp. 571–597. Chapman and Hall/CRC.



(a) Crossing Resolution (b) Angular Resolution (c) Total Resolution

(d) Crossing Resolution (e) Angular Resolution (f) Total Resolution

(g) Crossing Resolution (h) Angular Resolution (i) Total Resolution

(j) Crossing Resolution (k) Angular Resolution (l) Total Resolution

FIGURE 11: Different drawings of the 5th challenge graph given in the Graph Drawing 2017 contest produced by different variants of (a)–(c) the variant of our algorithm without restrictions on the aspect ratio, (d)–(f) the variant of our algorithm forced to maintain the input aspect ratio, (g)–(i) the algorithm by Argyriou et al. [21], and (j)–(l) the algorithm by Huang et al. [22]. The aesthetic criterion optimized by each variant is reported in the caption of its subfigure.

(a) Crossing Resolution (b) Angular Resolution (c) Total Resolution

(d) Crossing Resolution (e) Angular Resolution (f) Total Resolution

(g) Crossing Resolution (h) Angular Resolution (i) Total Resolution

(j) Crossing Resolution (k) Angular Resolution (l) Total Resolution

FIGURE 12: Different drawings of the 9th challenge graph given in the Graph Drawing 2017 contest produced by different variants of (a)–(c) the variant of our algorithm without restrictions on the aspect ratio, (d)–(f) the variant of our algorithm forced to maintain the input aspect ratio, (g)–(i) the algorithm by Argyriou et al. [21], and (j)–(l) the algorithm by Huang et al. [22]. Each variant was obtained by optimizing a different aesthetic criterion, which is named in the caption of each subfigure.