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, Jan Splett

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

Abstract. 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 have shown 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.

1 Introduction

In Graph Drawing, there exists a really rich literature and a wide range of techniques for drawing planar graphs; see, e.g., [11,28,34]. However, drawing a non-planar graph, and in particular when it does not have some special structure (e.g., degree restriction), is a difficult and challenging task, mainly due to the edge crossings that negatively affect the drawing's quality [39]. As a result, the established techniques are significantly fewer (e.g., crossing minimization heuristics [22,40], energy-based layout algorithms [20,24]); for an overview refer to [13,36,41].

In this context, Huang et al. [31,32] a decade ago introduced some important experimental evidence, 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, 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" [30,35,37], which focuses on graphs and their properties, when different constraints on the types of edges crossings are imposed; refer to [16] for a recent survey.

(a) (b)

Fig. 1: (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°.

Formally, the value of the minimum angle formed by any two crossing edges in a drawing is referred to as its *crossing resolution*; 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* graph or *RAC* graph, for short; see Figure 1. For these graphs, several results, mostly of theoretical nature, are known (refer to Section 2 for a short overview). Notably, RAC graph are sparse (they contain at most $4n - 10$ edges [15], where n denotes the number of vertices), while deciding whether a graph is RAC is NP-hard [4].

The latter result is already 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°. Also, the literature is significantly more limited, when restricting the crossing resolution to be smaller than 90°, as also evidenced by Section 2.

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 [20]. The first one is due to Huang et al. [33], while the second one is due to Argyriou et al. 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, 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.

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 [27] for computing large planar subgraphs. Since we also could not provide any theoretical guarantee on the expected quality of the produced drawings, we followed a more practical approach. We implemented our algorithm and the force-directed ones of [5] and [33], and we experimentally compared them on standard benchmark graphs. Our evaluation indicates that our method significantly outperforms the force-directed ones [5,33] in terms of crossing resolution, but this comes at the cost of slightly worse running time for large and dense graphs. Analogous results are obtained, when our algorithm and the ones of [5] and [33] are adjusted to maximize the *angular resolution* (i.e., the minimum value of the angle between any two adjacent edges [23]) or the *total resolution* (i.e., the minimum of the angular and the crossing resolution [5]).

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 $\Gamma(G)$.

Structure of the paper: The remainder of this paper is structured as follows. Section 2 overviews related works. Our algorithm is presented in detail in Section 3 and is experimentally evaluated against the ones of Huang et al. [33] and Argyriou et al. in Section 4. We conclude in Section 5 with open problems.

2 Related Work

As already mentioned, the study of the crossing resolution maximization problem has mainly focused on its optimal case, i.e., on the study of RAC graphs. An n-vertex RAC graph has at most $4n-10$ edges [15], while deciding whether a graph is RAC is NP-hard [4]. The maximally-dense RAC graphs are 1-planar [21], 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 [7,9]. Deciding, however, whether a 1-planar graph is RAC is NP-hard [8]. Note that the problem of finding RAC drawings has also been studied in the presence of bends [2,6,15,26] and by imposing restrictions on the degree [3], the structure [14] and the drawing [25,29] of the graph. The results are fewer, when the right-angle constraint is relaxed. Dujmovic et al. [19] proved that an n-vertex graph with crossing resolution at least α radians, has at most $(3n-6)\pi/\alpha$ edges. Corresponding density results are also known in the presence of bends [1,26].

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 force-directed techniques [13,20].

COWA is a system that supports conceptual web site traffic analysis [17]; its algorithmic core is a force-directed heuristic to compute simultaneous embeddings of two non-planar graphs with high crossing resolution. Didimo et al. [18] describe topology-driven force-directed heuristics to achieve good trade-offs in terms of number of edge crossings, crossing resolution, and geodesic edge tendency; the obtained drawings, however, are not straight-line. For straight-line drawings, Nguyen et al. [38] suggest a quadratic-program to increase the crossing angles of circular drawings. Of more general scope are the already mentioned force-directed algorithms of Argyriou et al. and Huang et al. [33].

3 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 G overlap in Γ_0, i.e., $c(\Gamma_0) > 0$. A circular drawing or a drawing obtained by applying a force-directed algorithm on 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 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. Then, a vertex of Γ_{i-1} is chosen arbitrarily at random based on the so-called *vertex-pool*, which may contain: (i) either all vertices of Γ_{i-1}, or (ii) a prespecified subset of the vertices of Γ_{i-1}, called *critical*.

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 Γ_{i-1}, if e and e' cross in Γ_{i-1} and the minimum angle that is formed at their crossing point is equal to $c(\Gamma_{i-1})$. The set of critical vertices of 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 [1]: By appropriately changing the location of a critical vertex or of a vertex 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 weighted random selection procedure, so that the vertices at distance i from the ones of the vertex-pool have higher weights than the corresponding ones at distance j in the graph, when $0 \leq i < j$. So, if the vertex-pool contains critical vertices, then the closer a vertex is to the critical vertices, the more likely it is to be chosen. Otherwise, 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

[1] 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 good solution, because it is simply very unlikely to select to move one of the vertices that define the crossing resolution.

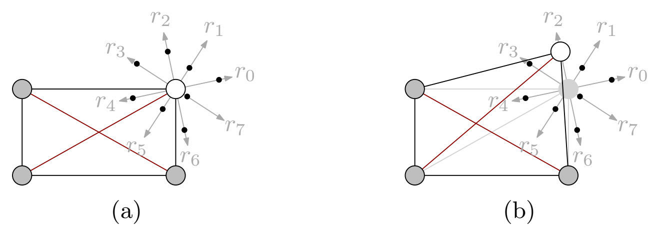


Fig. 2: Illustration of an iteration step of our algorithm: (a) The chosen vertex is the white one; the computed rays $r_0, \ldots, r_7$ have been rotated by $8°$; the black-colored points along these rays are points $\pi_0, \ldots, \pi_7$; among them, π_4 yields the best solution. (b) The resulting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

that the algorithm will get trapped to some local maxima after a number of iterations. So, special care is needed to avoid these bottlenecks, especially when the input graph is large. We discuss ways to avoid them later in this section.

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 described how to compute its new position in the next drawing.

Let v_i be the vertex of Γ_{i-1} that has been chosen to be moved at the i-th iteration. To compute the position of 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. 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 chose 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; point π_j is *feasible*, if the crossing resolution of the drawing obtained by placing vertex v_i at π_j and by keeping all other vertices of G in their positions in Γ_{i-1} is at least as large as the crossing resolution of Γ_{i-1}, and there is no vertex of Γ_{i-1} at π_j.

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., 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 v_i in Γ_i. The most natural is to chose 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 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.

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 behaviour. 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: (i) if during the last ζ iterations the crossing resolution has not been improved, then the vertex-pool becomes *wider* containing all the vertices, and the algorithm is executed with this vertex-pool for ζ' iterations; (ii) afterwards, the vertex-pool switches back to the critical vertices. While this approach turned out to be quite effective for medium-size graphs, for larger graphs, unfortunately, 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: (i) 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; (ii) afterwards, ρ, δ_{min} and δ_{max} switch back to this initial value. Of course, 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 [2,9].

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 naive 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.

If the algorithm had to compute all crossing points and the corresponding angles for each candidate position of each iteration, then it would not be useful in practice. Instead, we adopted a different approach, 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 points to move v_i. Let $e_0, \ldots, e_{d_i-1}$ be the edges incident to v_i, where $d_i = deg(v_i)$. Next, for each edge e_k with $k = 0, \ldots, d_i - 1$ we compute the crossings and the corresponding crossing angles of 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 v_i is at π_j. Let χ_j be the minimum crossing angle computed with this approach, when 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 $O(deg(v_i)m) = O(nm)$.

3.1 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 observations that we made during our experimental evaluation (see Section 4): Drawings with good crossing resolution tend to have bad aspect ratio and poor angular resolution. 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 [5]. 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 of Huang et al. [33] and Argyriou et al. [5], our algorithm can be also 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 outside the bounds.

4 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 of Argyriou et al. [5] and Huang et al. [33]. The implementations were in Java using the yFiles library [42]. The experiment was performed on a Linux laptop with four cores at 2.4 GHz and 8 GB RAM. As a test set for our experiment, we used the non-planar Rome graphs [12], which form a collection of around 8.100 benchmark graphs (commonly used for testing the efficiency of algorithms for drawing graphs).

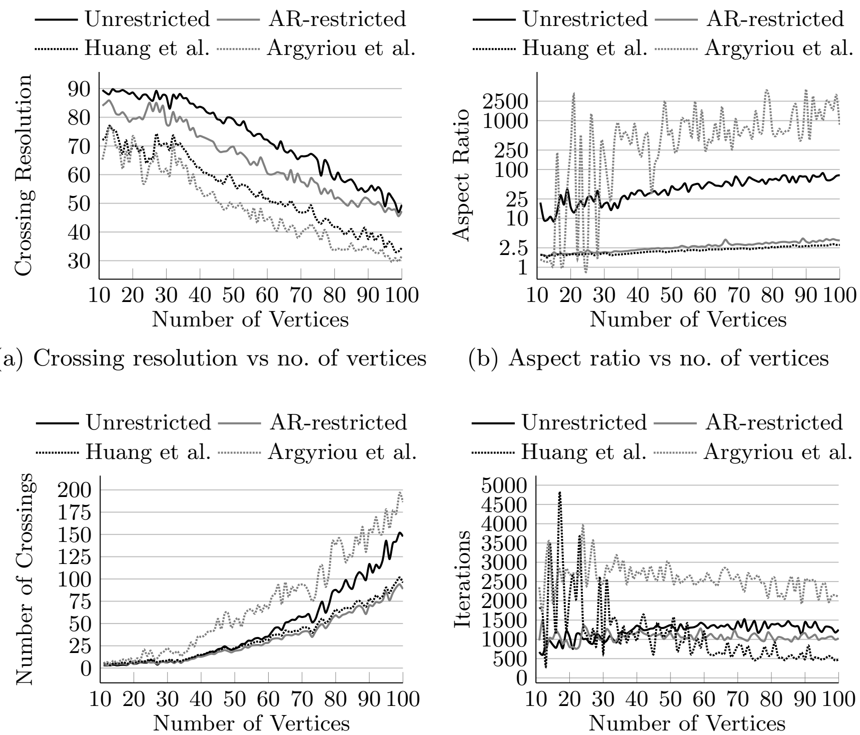
The experiment was performed as follows. Initially, each Rome graph was laid out using the SmartOrganic layouter of yFiles [42]. Starting from this layout, every graph was drawn with (i) our algorithm, (ii) our algorithm restricted not to violate the aspect ratio of the initial layout, and the force-directed algorithms (iii) by Argyriou et al. and (iv) by Huang et al. We compared the quality of the produced drawings based on the following aesthetic properties:

P.1. crossing resolution
P.2. total resolution
P.3. angular resolution
P.4. aspect ratio
P.5. number of crossings

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), for P.1, P.2 and P.3, we adjusted each of the algorithms to maximize exclusively the corresponding measures; see Figs. 3, 4 and 5. In our algorithm, this can be achieved by modifying appropriately the content of the vertex-pool (as we saw in Section 3.1), 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 are on average across different drawings with same number of vertices.

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),



(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

Fig. 3: Illustration of our experimental results on the crossing resolution.

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 outperform the corresponding spring forces, which try to keep the lengths of the edges short. Going back to our unrestricted algorithm, its behaviour is up to a certain degree expected, mainly due to the fact that there is no control on the lengths of the edges. 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) with the one of Huang et al.

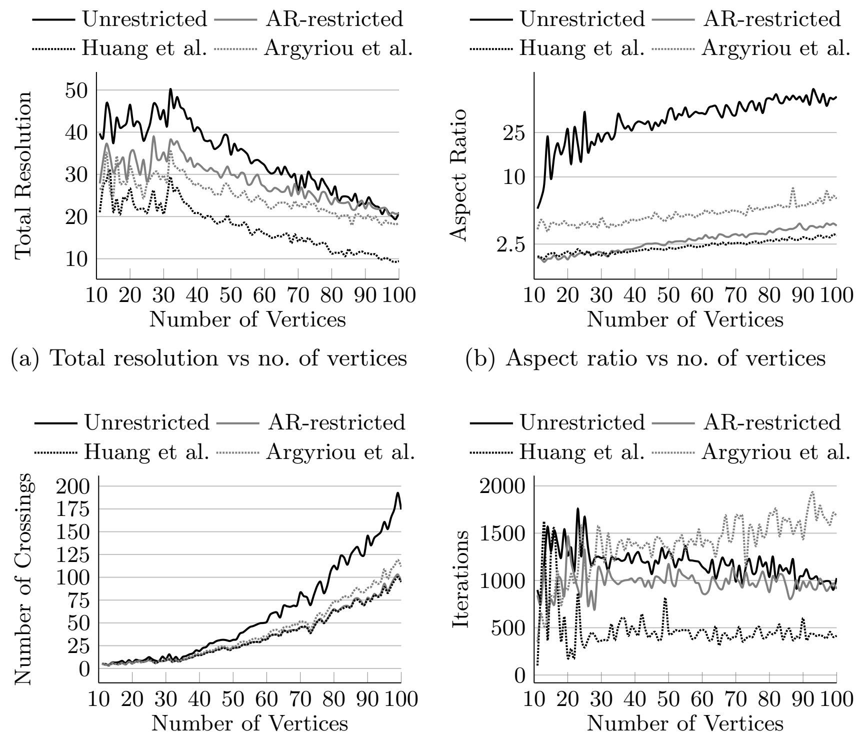
Regarding the number of crossings, we observe that the restricted variant of our algorithm and the force-directed algorithm of Huang et al. yield drawings with comparable number of crossings, which at the same time is significantly smaller than the corresponding number of crossings produced by the two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

A different behaviour 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 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 more restricted 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 could perform in order to converge was set to 100.000, but that limit was never reached. We observe that both force-directed algorithms seem to require a great amount of iterations to converge for small graphs, where a drawing with really good crossing resolution is possible. However, for bigger graphs the algorithm by Huang et al. requires the least amount of iterations. On the other hand, both the unrestricted and the restricted variant of our algorithm 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 the minimum of the crossing and of the angular resolution. 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. It is worth noting, 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 (even though there seems to be 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 aspect ratio) than the drawings produced by remaining algorithms; see Fig. 4b. More concretely, 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.

The restricted variant of our algorithm and the algorithm by Huang et al. yield drawings with the least number of crossings; see Fig. 4c. 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 seems to require the largest number of crossings, which turn out to be notably higher than the corresponding ones of the remaining algorithms of our experiment (especially for large graphs).



(a) Total resolution vs no. of vertices

(b) Aspect ratio vs no. of vertices

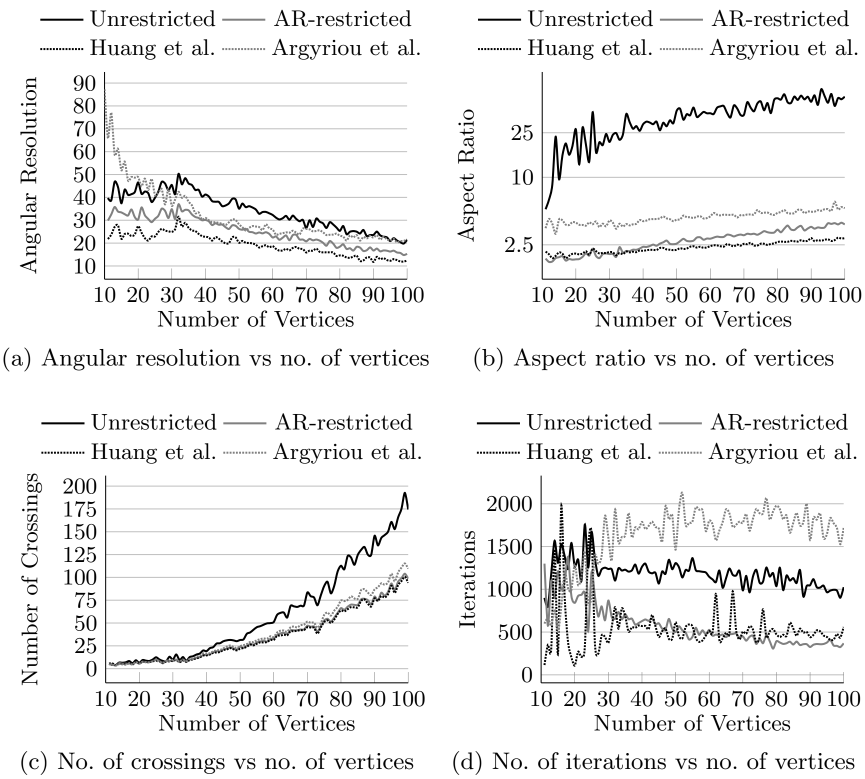
(c) No. of crossings vs no. of vertices

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

Fig. 4: Illustration of our experimental results on the total resolution.

On the negative side, both the unrestricted and the restricted variant of our algorithm require more iterations than the force-directed algorithm by Huang et al.; 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.



(a) Angular 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

Fig. 5: Illustration of our experimental results on the angular resolution.

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; 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.

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 state of the art force-directed algorithms, even when restricted to a given maximum aspect ratio. As a future direction, we plan to evaluate the performance of variants of our heuristic that compromise between even more aesthetic criteria.

References

1. E. Ackerman, R. Fulek, and C. D. Tóth. Graphs that admit polyline drawings with few crossing angles. *SIAM J. Discrete Math.*, 26(1):305–320, 2012. `doi:10.1137/100819564`.
2. P. Angelini, L. Cittadini, W. Didimo, F. Frati, G. Di Battista, M. Kaufmann, and A. Symvonis. On the perspectives opened by right angle crossing drawings. *J. Graph Algorithms Appl.*, 15(1):53–78, 2011. `doi:10.7155/jgaa.00217`.
3. P. Angelini, G. Di Battista, W. Didimo, F. Frati, S. Hong, M. Kaufmann, G. Liotta, and A. Lubiw. Large angle crossing drawings of planar graphs in subquadratic area. In A. Márquez, P. Ramos, and J. Urrutia, editors, *EGC*, volume 7579 of *LNCS*, pages 200–209. Springer, 2011. `doi:10.1007/978-3-642-34191-5_19`.
4. E. N. Argyriou, M. A. Bekos, and A. Symvonis. The straight-line RAC drawing problem is NP-hard. *J. Graph Algorithms Appl.*, 16(2):569–597, 2012. `doi:10.7155/jgaa.00274`.
5. E. N. Argyriou, M. A. Bekos, and A. Symvonis. Maximizing the total resolution of graphs. *Comput. J.*, 56(7):887–900, 2013. `doi:10.1093/comjnl/bxs088`.
6. K. Arikushi, R. Fulek, B. Keszegh, F. Moric, and C. D. Tóth. Graphs that admit right angle crossing drawings. *Comput. Geom.*, 45(4):169–177, 2012. `doi:10.1016/j.comgeo.2011.11.008`.
7. C. Bachmaier, F. J. Brandenburg, K. Hanauer, D. Neuwirth, and J. Reislhuber. Nic-planar graphs. *Discrete Applied Mathematics*, 232:23–40, 2017. `doi:10.1016/j.dam.2017.08.015`.
8. M. A. Bekos, W. Didimo, G. Liotta, S. Mehrabi, and F. Montecchiani. On RAC drawings of 1-planar graphs. *Theor. Comput. Sci.*, 689:48–57, 2017. `doi:10.1016/j.tcs.2017.05.039`.
9. F. J. Brandenburg, W. Didimo, W. S. Evans, P. Kindermann, G. Liotta, and F. Montecchiani. Recognizing and drawing ic-planar graphs. *Theor. Comput. Sci.*, 636:1–16, 2016. `doi:10.1016/j.tcs.2016.04.026`.
10. M. de Berg, O. Cheong, M. J. van Kreveld, and M. H. Overmars. *Computational geometry: algorithms and applications, 3rd Edition*. Springer, 2008.
11. H. de Fraysseix, J. Pach, and R. Pollack. How to draw a planar graph on a grid. *Combinatorica*, 10(1):41–51, 1990. `doi:10.1007/BF02122694`.
12. G. Di Battista and W. Didimo. Gdtoolkit. In R. Tamassia, editor, *Handbook on Graph Drawing and Visualization.*, pages 571–597. Chapman and Hall/CRC, 2013.
13. G. Di Battista, P. Eades, R. Tamassia, and I. G. Tollis. *Graph Drawing: Algorithms for the Visualization of Graphs*. Prentice-Hall, 1999.
14. W. Didimo, P. Eades, and G. Liotta. A characterization of complete bipartite RAC graphs. *Inf. Process. Lett.*, 110(16):687–691, 2010. `doi:10.1016/j.ipl.2010.05.023`.
15. W. Didimo, P. Eades, and G. Liotta. Drawing graphs with right angle crossings. *Theor. Comput. Sci.*, 412(39):5156–5166, 2011. `doi:10.1016/j.tcs.2011.05.025`.
16. W. Didimo, G. Liotta, and F. Montecchiani. A survey on graph drawing beyond planarity. *CoRR*, abs/1804.07257, 2018. `arXiv:1803.03705`.
17. W. Didimo, G. Liotta, and S. A. Romeo. Graph visualization techniques for conceptual web site traffic analysis. In *IEEE PacificVis*, pages 193–200. IEEE Computer Society, 2010. `doi:10.1109/PACIFICVIS.2010.5429593`.
18. W. Didimo, G. Liotta, and S. A. Romeo. Topology-driven force-directed algorithms. In U. Brandes and S. Cornelsen, editors, *Graph Drawing*, volume 6502 of *LNCS*, pages 165–176. Springer, 2010. `doi:10.1007/978-3-642-18469-7_15`.

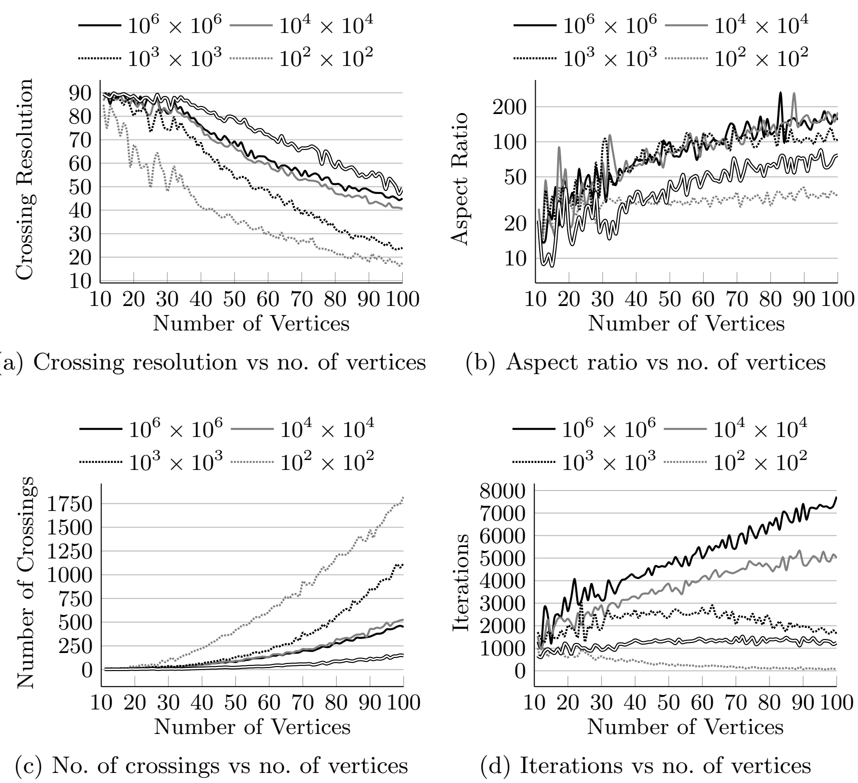
19. V. Dujmovic, J. Gudmundsson, P. Morin, and T. Wolle. Notes on large angle crossing graphs. *Chicago J. Theor. Comput. Sci.*, 2011, 2011. URL: `http://cjtcs.cs.uchicago.edu/articles/CATS2010/4/contents.html`.
20. P. Eades. A heuristic for graph drawing. *Congressus Numerantium*, 42:149–160, 1984.
21. P. Eades and G. Liotta. Right angle crossing graphs and 1-planarity. *Discrete Applied Mathematics*, 161(7-8):961–969, 2013. `doi:10.1016/j.dam.2012.11.019`.
22. P. Eades and N. C. Wormald. Edge crossings in drawings of bipartite graphs. *Algorithmica*, 11(4):379–403, 1994. `doi:10.1007/BF01187020`.
23. M. Formann, T. Hagerup, J. Haralambides, M. Kaufmann, F. T. Leighton, A. Symvonis, E. Welzl, and G. J. Woeginger. Drawing graphs in the plane with high resolution. *SIAM J. Comput.*, 22(5):1035–1052, 1993. `doi:10.1137/0222063`.
24. T. M. J. Fruchterman and E. M. Reingold. Graph drawing by force-directed placement. *Softw., Pract. Exper.*, 21(11):1129–1164, 1991. `doi:10.1002/spe.4380211102`.
25. E. D. Giacomo, W. Didimo, P. Eades, and G. Liotta. 2-layer right angle crossing drawings. *Algorithmica*, 68(4):954–997, 2014. `doi:10.1007/s00453-012-9706-7`.
26. E. D. Giacomo, W. Didimo, G. Liotta, and H. Meijer. Area, curve complexity, and crossing resolution of non-planar graph drawings. *Theory Comput. Syst.*, 49(3):565–575, 2011. `doi:10.1007/s00224-010-9275-6`.
27. O. Goldschmidt and A. Takvorian. An efficient graph planarization two-phase heuristic. *Networks*, 24(2):69–73, 1994. URL: `https://doi.org/10.1002/net.3230240203`, `doi:10.1002/net.3230240203`.
28. C. Gutwenger and P. Mutzel. Planar polyline drawings with good angular resolution. In S. Whitesides, editor, *Graph Drawing*, volume 1547 of *LNCS*, pages 167–182. Springer, 1998. `doi:10.1007/3-540-37623-2_13`.
29. S. Hong and H. Nagamochi. Testing full outer-2-planarity in linear time. In E. W. Mayr, editor, *WG*, volume 9224 of *LNCS*, pages 406–421. Springer, 2015. `doi:10.1007/978-3-662-53174-7_29`.
30. S. Hong and T. Tokuyama. Algorithmics for beyond planar graphs. NII Shonan Meeting Seminar 089, November 27 - December 1 2016.
31. W. Huang. Using eye tracking to investigate graph layout effects. In S. Hong and K. Ma, editors, *APVIS*, pages 97–100. IEEE Computer Society, 2007. `doi:10.1109/APVIS.2007.329282`.
32. W. Huang, P. Eades, and S. Hong. Larger crossing angles make graphs easier to read. *J. Vis. Lang. Comput.*, 25(4):452–465, 2014. `doi:10.1016/j.jvlc.2014.03.001`.
33. W. Huang, P. Eades, S. Hong, and C. Lin. Improving multiple aesthetics produces better graph drawings. *J. Vis. Lang. Comput.*, 24(4):262–272, 2013. `doi:10.1016/j.jvlc.2011.12.002`.
34. G. Kant. Drawing planar graphs using the canonical ordering. *Algorithmica*, 16(1):4–32, 1996. `doi:10.1007/BF02086606`.
35. M. Kaufmann, S. Kobourov, J. Pach, and S. Hong. Beyond planar graphs: Algorithmics and combinatorics. Dagstuhl Seminar 16452, November 6-11 2016.
36. M. Kaufmann and D. Wagner, editors. *Drawing Graphs, Methods and Models*, volume 2025 of *LNCS*. Springer, 2001. `doi:10.1007/3-540-44969-8`.
37. G. Liotta. Graph drawing beyond planarity: Some results and open problems. SoCG Week, Invited talk, July 4th 2017.
38. Q. H. Nguyen, P. Eades, S. Hong, and W. Huang. Large crossing angles in circular layouts. In U. Brandes and S. Cornelsen, editors, *Graph Drawing*, volume 6502 of *LNCS*, pages 397–399. Springer, 2010. `doi:10.1007/978-3-642-18469-7_40`.

39. H. C. Purchase. Effective information visualisation: a study of graph drawing aesthetics and algorithms. *Interacting with Computers*, 13(2):147–162, 2000. doi:10.1016/S0953-5438(00)00032-1.
40. K. Sugiyama, S. Tagawa, and M. Toda. Methods for visual understanding of hierarchical system structures. *IEEE Trans. Systems, Man, and Cybernetics*, 11(2):109–125, 1981. doi:10.1109/TSMC.1981.4308636.
41. R. Tamassia, editor. *Handbook on Graph Drawing and Visualization*. Chapman and Hall/CRC, 2013.
42. R. Wiese, M. Eiglsperger, and M. Kaufmann. *yFiles* - visualization and automatic layout of graphs. In *Graph Drawing Software*, pages 173–191. Springer, 2004. doi:10.1007/978-3-642-18638-7_8.

Appendix

A Experiments on Grid Drawings

In addition to the experiments described in Section 4, we also evaluated how our algorithm performs, if we restrict its vertices to be placed on grid coordinates whilst avoiding vertices to overlap with other vertices or edges. Further, we restricted the grid size to (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 set for this experiment was again the set of non-planar Rome graphs [12], but we computed a different initial since our algorithm requires its input drawing to maintain its invariants (that is, vertices must be on the grid). More precisely, we computed a random grid layout of each graph where vertices to overlap with other vertices or edges. For each grid size, we computed a layout with the variant of our algorithm focusing on crossing resolution. Fig. 6 summarizes the results.



(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

Fig. 6: Illustration of our experimental results on the crossing resolution with grid restriction. The double line shows the results of our unrestricted algorithm from Section 4.

Regarding the crossing resolution, we can observe that with increasing grid size, we could achieve better crossing resolution; see Fig. 6a. More precisely, the grid of size $10^2 \times 10^2$ was too restrictive for the vast majority of graphs which often resulted in the initial layout to be accepted as the best solution for larger graphs. Starting from grid size $10^3 \times 10^3$, we could observe, for small graphs, the performance was close to the performance of the unrestricted version of our algorithm (double line in Fig. 6a). For grid size $10^3 \times 10^3$, the performance declined for the larger graphs in the test set and we again could observe that for some of them the algorithm was not able to improve the initial layout. For grid size $10^4 \times 10^4$, the algorithm performed only about $10°$ worse than the unrestricted version of our algorithm which could only slightly be improved by increasing the grid size to $10^6 \times 10^6$.

The aspect ratio of computed drawings was more or less the same for all grid size with the exception of size $10^2 \times 10^2$ and always about twice the aspect ratio of the unrestricted variant of our algorithm; see Fig. 6b. For grid size $10^2 \times 10^2$, this may again be explained by the fact, that for the majority of graphs, the initial layout could not be improved.

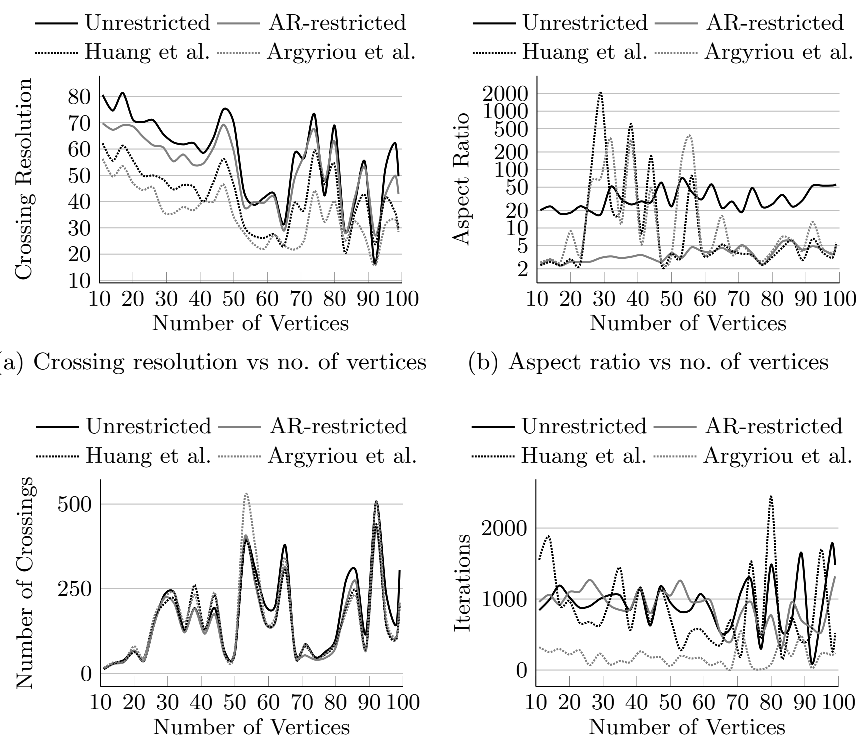
The number of crossings increased with the restriction on the size of the grid; see Fig. 6c. As with the crossing resolution, there is clear differences between grid sizes $10^2 \times 10^2$, $10^3 \times 10^3$ and $10^4 \times 10^4$ whereas there is only a slight improvement from grid size $10^4 \times 10^4$ to $10^6 \times 10^6$ which still uses twice as many crossings as our unrestricted variant. This can partially be explained with the different choice of the initial drawing.

Finally, we observe that the number of iterations needed for convergence for grid size $10^3 \times 10^3$ was already twice as much as for our unrestricted algorithm; see Fig. 6d. From there on, the number of iterations for convergence increases with the grid size. Note that grid size $10^6 \times 10^6$ needed about 1000 to 2000 iterations more than grid size $10^4 \times 10^4$ even though the final layout was only marginally better as discussed prior. The curve for $10^2 \times 10^2$ highlights that this grid size was too restrictive for the vast majority of graphs which often resulted in the initial layout to be accepted as the best solution for larger graphs.

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 grid drawings takes much longer than in the unrestricted version (up to five times as long for grid size $10^4 \times 10^4$ or eight times as long for grid size $10^6 \times 10^6$). In order to improve the performance of our algorithm in this restricted variant, it may be interesting to see how much the choice of the initial drawing affects the resulting crossing resolution. Further experiments in this direction are needed.

B Experiments on the AT&T Graph Test Set

We repeated our experimental evaluation on the crossing resolution, total resolution and angular resolution without grid constraint on a second test set of graphs, the set of non-planar AT&T graphs, which form a collection of 424 benchmark graphs (also known as Graph Catalog and North graphs; available at `http://graphdrawing.org/data`). The corresponding results are illustrated in Figs. 7, 8 and 9. In general we observed that the variance of results is much larger than in the experiments on the Rome graphs. This manifests in spikes of large magnitude in the illustrations of results and indicates that the structural properties of graphs in this second test set varies vastly between different graph sizes.

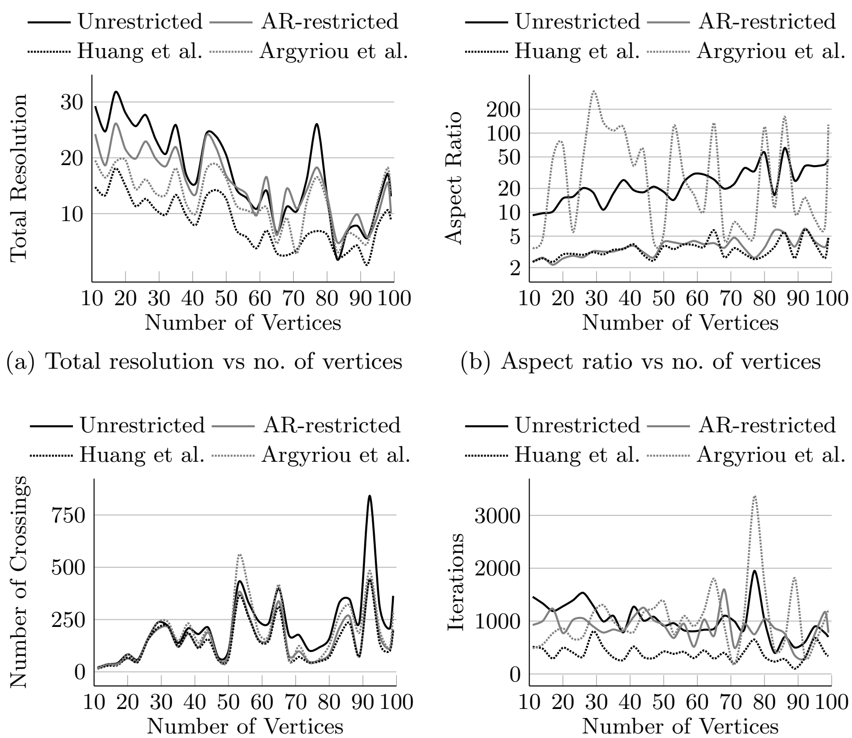


(a) Crossing resolution vs no. of vertices (b) Aspect ratio vs no. of vertices

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

Fig. 7: Experimental results for the crossing resolution experiment on the North graph testset.

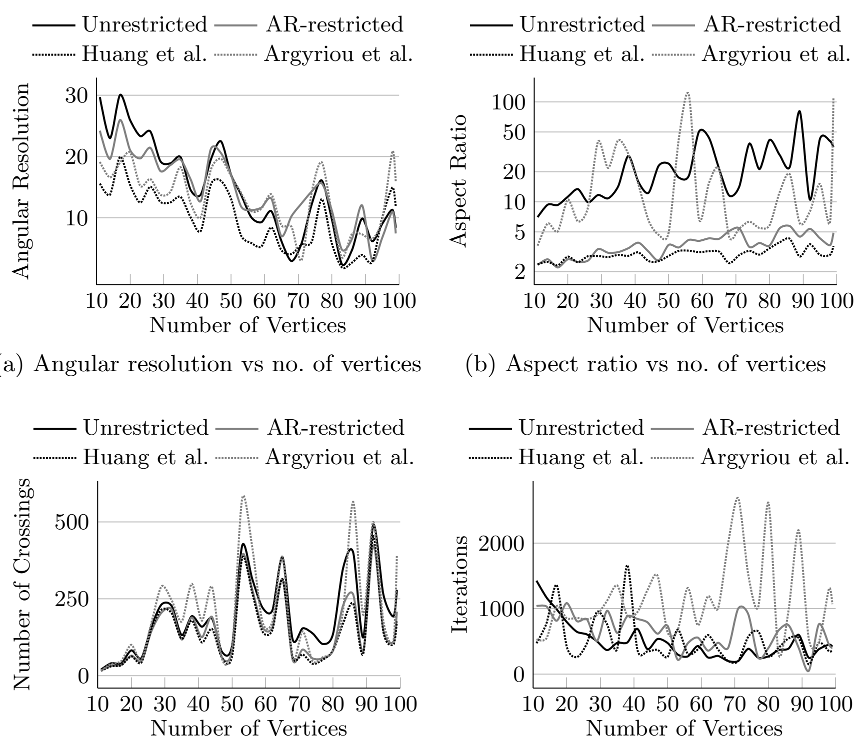
For the crossing resolution we observed that both the unrestricted and the aspect ratio restricted variants of our algorithm again outperformed the two evaluated force directed approachs; see Fig. 7a. Remarkable is the synchronous behaviour of all four algorithms with respect to the crossing resolution on different graph sizes such that the curves are nearly parallel. By all these results, we can classify the graphs into "hard" or "easy" graphs with respect to the crossing resolution maximization; in particular graphs between 50 and 70 vertices appear to be harder to improve than graphs between 70 and 80 vertices. With respect to the aspect ratio of produced drawings we observe that while our algorithms show a slight increase with the number of vertices, the behaviour for both force directed algorithms appears to be quite unstable resulting in a large variance. Again the restricted variant of our algorithm and the two force directed approaches 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 number of crossings; see Fig. 7c. In terms of the number of iterations, we observe that surprisingly the algorithm of Argyriou et al. converges in the least amout of iterations throughout the test set whereas the remaining three algorithms behave nearly the same; see Fig. 7d.



(a) Total 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

Fig. 8: Experimental results for the total resolution experiment on the North graph testset.

In the total resolution experiment, for smaller graphs, we observed similar results as in the experiment on the Rome graphs, that is, our unrestricted algorithm achieves best total resolution followed by the restricted variant and then the algorithm by Argyriou et al.; see Fig. 8a. For larger graphs, however, these three algorithms achieve similar results while still outperforming the algorithm by Huang et al. The results for the aspect ratio and number of crossings are similar to those of the crossing resolution experiment, with the exception of the fact that the algorithm of Huang et al. performs more stable with respect to the aspect ratio; see Figs. 8b and 8c. With respect to the number of iterations our two algorithms and the one by Argyriou et al. show similar behaviour needing more iterations than the algorithm by Huang et al.; see Fig. 8d. Observe that the number of iterations does not seem to correlate with the number of vertices.



(a) Angular 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

Fig. 9: Experimental results for the angular resolution experiment on the North graph testset.

In the angular resolution experiments we again obtain a not-so-clear picture concerning the ranking of the algorithms, especially for higher number of vertices the ranking varies; see Fig. 9a. Only the algorithm by Huang et al. seems to be mostly at the last rank. Concerning the aspect ratio we see very good behaviour for our restricted variant and the algorithm by Huang et al. while the remaining two algorithms show large variance and much worse values; see Fig. 9b. For the number of crossings, we again observe that all algorithms achieve similar values, however, our unrestricted algorithm and the algorithm by Argyriou et al. achieve slightly higher values for larger graphs; see Fig. 9c. Finally, both our algorithms and the one by Huang et al. need a similar number of iterations for convergence which is lower than the one by Argyriou et al.; see Fig. 9d.

C Graph Drawing Contest 2017 Graphs

We give a comparison of our new approach to the performances of the clear winner "CoffeeVM" of last year's graph drawing contest[2] on the crossing angle maximization and our previous team "TuebingenMidnight" in Table 1. Note that in the contest the teams had only one hour to compute a layout for all 15 contest graphs.

Table 1: Results for the Graph Drawing Contest Graphs.

Graph	CoffeeVM	Our New Approach	TuebingenMidnight
1	90	89.78	77
2	88.23	88.7	42
3	90	89.95	89
4	88.97	89.05	89
5	80.4	86.96	30
6	90	89.72	78
7	56.537	63.62	34
8	84.95	89.28	61
9	59.885	88.2	9
10	20.978	23.72	4
11	46.684	72	6
12	36.47	35.86	5
13	25.456	33.68	4
14	33.52	43.08	5
15	20.512	29.18	4

For all the graphs, our results are (sometimes considerably) better than or about the same as the contest winner's.

[2] `http://www.graphdrawing.de/contest2017/results.html`