The Computer Journal Advance Access published June 27, 2012

doi:10.1093/comjnl/bxs088

Maximizing the Total Resolution of Graphs

E.N. ARGYRIOU[1], M.A. BEKOS[2] AND A. SYMVONIS[1,*]

[1]*School of Applied Mathematical and Physical Sciences, National Technical University of Athens, 15780 Zografou, Athens, Greece*
[2]*Institute for Informatics, University of Tübingen, Sand 13, 72076 Tübingen, Germany*
[*]*Corresponding author: symvonis@math.ntua.gr*

A major factor affecting the readability of a graph drawing is its resolution. In the graph drawing literature, the resolution of a drawing is either measured based on the angles formed by consecutive edges incident on a common vertex (angular resolution) or by the angles formed at edge crossings (crossing resolution). In this paper, we introduce the notion of 'total resolution', that is, the minimum of the angular and crossing resolution. To the best of our knowledge, this is the first time where the problem of maximizing the total resolution of a graph over all its drawings is studied. The main contribution of the paper consists of drawings of asymptotically optimal total resolution for complete graphs (circular drawings) and for complete bipartite graphs (two-layered drawings). In addition, we present and experimentally evaluate a force-directed-based algorithm that constructs drawings of large total resolution.

Keywords: graph drawing; angular; crossing and total resolution

Received 23 February 2012; revised 24 April 2012
Handling editor: Iain Stewart

1. INTRODUCTION

Graphs are widely used to depict relations between objects. There exist several criteria that have been used to judge the quality of a graph drawing [1, 2]. From a human point of view, it is necessary to confront drawings that are easy-to-read, i.e. they should nicely convey the structure of the objects and their relationships. From an algorithmic point of view, the quality of a drawing is usually evaluated by some objective function and the main task of a graph drawing algorithm is to determine a drawing that minimizes or maximizes the specific objective function. Various such functions have been studied by the graph drawing community, among them, the number of crossings among pairs of edges, the number of edge bends, the maximum edge length, the total area occupied by the drawing and so on [2, 3].

Over the past years, a significant amount of research effort has been devoted to the problem of reducing the number of crossings. This is reasonable, since it is commonly accepted that edge crossings may negatively affect the quality of a drawing. Toward this direction, there also exist eye-tracking experiments that confirm the negative impact of edge crossings on the human understanding of a graph drawing [4–6]. However, the computational complexity of the edge crossing minimization problem, which is $\mathcal{NP}$-complete in general [7], implies that the computation of high-quality drawings of dense graphs is difficult to achieve.

Apart from the edge crossings, another undesired property that may negatively influence the readability of a drawing is the presence of edges that are too close to each other, especially if these edges are adjacent. Thus, maximizing the angles among incident edges becomes an important esthetic criterion, since there is some correlation between the involved angles and the visual distinctiveness of the edges.

Motivated by the cognitive experiments by Huang [8] and Huang *et al.* [9] that indicate that the negative impact of an edge crossing is reduced in the case where the crossing angle is $>70°$, we study a new graph drawing scenario in which both the *angular* and *crossing resolution*[1] are taken into account in order to produce a straight-line drawing of a given graph. To the best of our knowledge, this is the first attempt, where both the angular and crossing resolution are combined to produce

[1]The term *angular resolution* denotes the smallest angle formed by two adjacent edges incident on a common vertex, whereas the term *crossing resolution* refers to the smallest angle formed by a pair of crossing edges.

drawings with high resolution. We show how to construct drawings that asymptotically maximize the minimum of the angular and crossing resolution for the classes of complete and complete bipartite graphs (Section 3). We also present a more practical, force-directed-based algorithm that constructs drawings of large angular and crossing resolution (Section 4).

1.1. Previous work

Formann *et al.* [10] were the first to study the angular resolution of straight-line drawings. They proved that deciding whether a graph of maximum degree d admits a drawing of angular resolution $2\pi/d$ (i.e. the obvious upper bound) is $\mathcal{NP}$-hard. They also proved that several types of graphs of maximum degree d (including planar graphs, complete graphs, hypercubes, multidimensional meshes and tori) have angular resolution $\Theta(1/d)$. Malitz and Papakostas [11] proved that any planar graph of maximum degree d admits a planar straight-line drawing with angular resolution $\Omega(1/7^d)$. Garg and Tamassia [12] showed a continuous trade-off between the area and the angular resolution of planar straight-line drawings. Gutwenger and Mutzel [13] gave a linear time and space algorithm that constructs a planar polyline grid drawing of a connected planar graph with n vertices and maximum degree d on a $(2n-5) \times (\frac{3}{2}n - \frac{7}{2})$ grid with at most $5n - 15$ bends and minimum angle $> 2/d$. Bodlaender and Tel [14] showed that planar graphs with angular resolution at least $\pi/2$ are rectilinear.

A graph is called a *right angle crossing* (or *RAC* for short) graph if it admits a polyline drawing in which every pair of crossing edges intersects at a right angle. Didimo *et al.* [15] were the first to study RAC drawings and they showed that any straight-line RAC drawing with n vertices has at most $4n - 10$ edges. Arikushi *et al.* [16] presented bounds on the number of edges of polyline RAC drawings with at most one or two bends per edge. Angelini *et al.* [17] showed that there are acyclic planar digraphs not admitting straight-line upward RAC drawings and that the corresponding decision problem is $\mathcal{NP}$-hard. They also constructed digraphs whose straight-line upward RAC drawings require exponential area. Argyriou *et al.* [18] proved that the problem of deciding whether a given graph admits a straight-line RAC drawing (i.e. the upwardness requirement is relaxed) is $\mathcal{NP}$-hard. Di Giacomo *et al.* [19] presented trade-offs between the crossing resolution, the maximum number of bends per edges and the area of a drawing. Didimo *et al.* [20] presented a characterization of complete bipartite graphs that admit straight-line RAC drawings. Di Giacomo *et al.* [21] presented upper and lower bounds to the crossing resolution of complete graphs. van Kreveld [22] studied how much better (in terms of the area required, edge-length and angular resolution) an RAC drawing of a planar graph can be than any planar drawing of the same graph. Eades and Liotta [23] proved that a *maximally dense RAC graph* (i.e. $|E| = 4|V| - 10$) is also 1-planar, i.e. it admits a drawing in which every edge is crossed at most once. Dujmovic *et al.* [24] studied α *Angle Crossing* (or α*AC* for short) graphs, in which the smallest angle formed by an edge crossing is at least α. For this class of graphs, they proved upper and lower bounds for the number of edges. Angelini *et al.* [25] presented algorithms for computing non-planar drawings of planar graphs that require subquadratic area, assuming that the angles formed by edge crossings are arbitrarily close to 90° and the maximum number of bends per edge is bounded by a fixed constant.

Force-directed methods are commonly used for drawing graphs [26–29]. In such a framework, a graph is treated as a physical system with forces acting on it. Then, a good configuration or drawing can be obtained from an equilibrium state of the system. An overview of force-directed methods and their variations can be found in the graph drawing books [2, 3]. Brandenburg *et al.* [30] presented an experimental comparison of five well-known force-directed and randomized graph drawing algorithms. Bertault [31] presented a force-directed method that preserves the edge crossing properties of an input layout, i.e. two edges cross in the produced drawing if and only if they cross in the input drawing as well. More recently, Lin and Yen [32] presented a force-directed method that leads to drawings with high angular resolution. In their work, pairs of edges incident on a common vertex are modeled as charged springs, that repel each other, guaranteeing the absence of any overlap among edges incident on a common vertex. Didimo *et al.* [33] proposed a framework that combines force-directed and planarization-based approaches to obtain polyline drawings of non-planar graphs with a small number of crossings, high crossing resolution and good geodesic edge tendency.[2] In the context of polyline drawings again, crossing resolution and geodesic edge tendency are also the main optimization criteria of a force-directed algorithm of Didimo *et al.* [34] that computes a simultaneous embedding of two input graphs in order to support website traffic analysis by means of graph visualization techniques.

The rest of this paper is structured as follows. In Section 2, we introduce preliminary notions and notation that are used in the rest of this paper. In Sections 3, we present theoretical results for the classes of complete bipartite and complete graphs. In Section 4, we present a force-directed algorithm, which results in the drawings of large angular and crossing resolution. We conclude in Section 6 with open problems and future work.

A preliminary version of this work has appeared in [35]. At the same time, Huang *et al.* [36] have independently published a force-directed algorithm, which uses a subset of the forces utilized in our algorithm and different functions to control the magnitude of these forces.

[2]For an edge $e = (u, v)$ with bends, the term geodesic edge tendency refers to the maximum distance from a point of e to the straight-line segment connecting u and v.

2. PRELIMINARY NOTIONS AND NOTATION

Let $G = (V, E)$ be an undirected graph. Given a drawing $\Gamma(G)$ of G, we denote by $p_u = (x_u, y_u)$ the position of vertex $u \in V$ on the plane. The vector from p_u to p_v which has unit length is denoted by $\overrightarrow{p_u p_v}$, where $u, v \in V$. The degree of vertex $u \in V$ is denoted by $d(u)$. Let also $d(G) = \max_{u \in V} d(u)$ be the degree of the graph.

Given a pair of points $q_1, q_2 \in \mathbb{R}^2$, with a slight abuse of notation, we denote by $\|q_1 - q_2\|$ the Euclidean distance between q_1 and q_2. We refer to the line segment defined by q_1 and q_2 as $\overline{q_1 q_2}$.

Let $\overrightarrow{\alpha}$ and $\overrightarrow{\gamma}$ be two vectors. The vector that bisects the angle between $\overrightarrow{\alpha}$ and $\overrightarrow{\gamma}$ is $\overrightarrow{\alpha}/\|\overrightarrow{\alpha}\| + \overrightarrow{\gamma}/\|\overrightarrow{\gamma}\|$. We denote by $\texttt{Bsc}(\overrightarrow{\alpha}, \overrightarrow{\gamma})$ the corresponding unit length vector. Given a vector $\overrightarrow{\beta}$, we refer to the unit length vector that is perpendicular to $\overrightarrow{\beta}$ and precedes it in the clockwise direction, as $\texttt{Perp}(\overrightarrow{\beta})$. Some of our proofs use the following elementary geometric properties:

$$\tan(\omega_1 \pm \omega_2) = \frac{\tan\omega_1 \pm \tan\omega_2}{1 \pm (-\tan\omega_1 \cdot \tan\omega_2)}, \tag{1}$$

$$\tan(\omega/2) = \frac{\sin\omega}{1 + \cos\omega}, \tag{2}$$

$$\omega \in \left(0, \frac{\pi}{2}\right) \Rightarrow \tan\omega > \omega. \tag{3}$$

3. DRAWINGS WITH OPTIMAL TOTAL RESOLUTION FOR COMPLETE AND COMPLETE BIPARTITE GRAPHS

In this section, we first define formally the total resolution of a drawing. Then, we present two methods for obtaining drawings of asymptotically optimal total resolution for complete and complete bipartite graphs. In particular, we prove that (i) any complete graph K_n admits a circular drawing of total resolution $O(1/n)$, which is asymptotically optimal, and, (ii) any complete bipartite graph $K_{n,m}$ admits a layered drawing of total resolution $O(1/\max\{n, m\})$, which is also asymptotically optimal. Obviously, the problem of determining the maximum total resolution for the case of planar straight-line drawings of planar graphs is equivalent to the corresponding problem of determining the maximum angular resolution, which is a well-studied problem in the graph drawing literature. Therefore, we concentrate our study on non-planar classes of graphs. As a first step toward solving the total resolution maximization problem on arbitrary non-planar graphs, we study the intuitively easier cases of complete and complete bipartite graphs, for which we seek to obtain circular and two-layered drawings of maximum total resolution, respectively. Note that such drawings are common for the visualization of complete and complete bipartite graphs. The formal definition of the term 'total resolution' is given below.

DEFINITION 3.1. *The total resolution of a drawing is defined as the minimum of its angular and crossing resolution.*

We first consider the case of complete graphs. Let $K_n = (V, E)$ be a complete graph, where $V = \{u_0, u_1, \ldots, u_{n-1}\}$ and $E = V \times V$. Our aim is to construct a circular drawing of K_n of maximum total resolution. Our approach is constructive and quite common when dealing with complete graphs. A similar construction has been given by Formann *et al.* [10] for obtaining optimal drawings of complete graphs, in terms of angular resolution. Consider a circle $\mathcal{C}$ of radius $r_c > 0$ centered at $(0, 0)$ and inscribe a regular n-polygon $\mathcal{Q}$ on $\mathcal{C}$. In our construction, the vertices of K_n coincide with the vertices of $\mathcal{Q}$. Without loss of generality, we further assume that $u_0, u_1, \ldots, u_{n-1}$ appear in this order in the counter-clockwise direction around $(0, 0)$, as illustrated in Fig. 1a.

THEOREM 3.1. *A complete graph K_n admits a circular drawing of total resolution $O(1/n)$, which is asymptotically optimal.*

Proof. We prove that the angular resolution of the presented drawing of K_n is π/n, whereas its crossing resolution is $2\pi/n$. First, observe that the length arc of circle $\mathcal{C}$ that connects two consecutive vertices u_i and $u_{(i+1) \bmod n}$ is equal to $2\pi r_c/n$ for each $i = 0, 1, \ldots, n-1$. Therefore, the angular resolution of the drawing is π/n, as desired. Now let $e_i = (u_i, u_{i'})$ and $e_j = (u_j, u_{j'})$ be two crossing edges. Without loss of generality, we assume that $i < j < i' < j'$, as in Fig. 1a. The crossing of e_i and e_j defines two angles ϕ_c and ϕ'_c such that $\phi_c + \phi'_c = \pi$. In Fig. 1a, ϕ_c is exterior to the triangle formed by the crossing of e_i and e_j and the vertices u_j and $u_{i'}$ (refer to the dark-gray triangle of Fig. 1a). Therefore: $\phi_c = (j' - i')(\pi/n) + (j - i)(\pi/n)$. Similarly, $\phi'_c = (i' - j)(\pi/n) + (n - (j' - i))(\pi/n)$. In the case in which $j = (i + 1) \mod n$ and $j' = (i' + 1) \mod n$ [i.e. the vertices u_i ($u_{i'}$, respectively) and u_j ($u_{j'}$, respectively) are consecutive], angle ϕ_c receives its minimum value, which is equal to $2\pi/n$. Similarly, we can prove that the minimum value of ϕ'_c is also $2\pi/n$. This establishes that the crossing resolution of the drawing is $2\pi/n$. Note that the same upper bound for the crossing resolution of K_n has been independently proved by Di Giacomo *et al.* [21]. □

We now proceed to consider the class of complete bipartite graphs. Since an n-vertex complete bipartite graph is a subgraph of an n-vertex complete graph, a drawing of $\Theta(1/n)$ total resolution for the complete bipartite graph can be derived by the optimal drawing for the complete graph. However, if the vertices of the graph must have integer coordinates (i.e. we restrict ourselves to grid drawings), few results are known regarding the area needed for such a drawing. An upper bound of $O(n^3)$ area can be derived by Bárány and Tokushige [37]. This motivates us to separately study the class of complete bipartite graphs, since we can drastically improve this bound. Note that

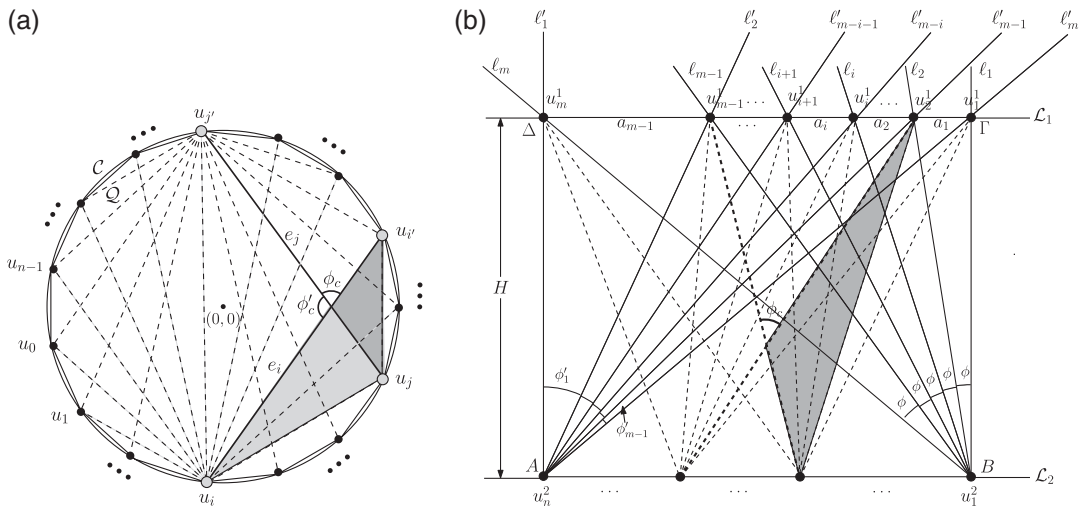

FIGURE 1. Illustrations of our constructions. (**a**) A circular drawing of K_n. (**b**) A 2-layered drawing of $K_{m,n}$.

trade-offs between (angular or crossing) resolution and area have been studied in the past. Malitz and Papakostas [11] showed that there exist graphs that always require exponential area for straight-line embeddings maintaining good angular resolution. The claim remains true, if circular arc edges are used instead of straight lines [38]. More recently, Angelini *et al.* [17] constructively showed that there exist graphs whose straight-line upward RAC drawings require exponential area.

Let $K_{m,n} = (V_1 \cup V_2, E)$ be a complete bipartite graph, where $V_1 = \{u^1_1, u^1_2, \ldots, u^1_m\}$, $V_2 = \{u^2_1, u^2_2, \ldots, u^2_n\}$ and $E = V_1 \times V_2$. Without loss of generality, we assume that $m \geq n$. As already stated, we seek to obtain two-layered drawings in which the vertices of V_1 and V_2 lie on two parallel lines, say $\mathcal{L}_1$ and $\mathcal{L}_2$, respectively. For the sake of simplicity, we assume that both $\mathcal{L}_1$ and $\mathcal{L}_2$ are horizontal. We first provide a geometric construction according to which the vertices are not required to be on grid points and then we show how to convert it into a grid drawing.

Initially, we consider a square $\mathcal{R} = AB\Gamma\Delta$, whose top and bottom sides coincide with $\mathcal{L}_1$ and $\mathcal{L}_2$, respectively (Fig. 1b). Let H be the height (and width) of $\mathcal{R}$. According to our approach, the vertices of V_1 (V_2, respectively) reside alongside $\Gamma\Delta$ (AB, respectively) of $\mathcal{R}$. To specify the exact positions of the vertices $u^1_1, u^1_2, \ldots, u^1_m$ alongside $\Gamma\Delta$, we first construct a bundle of m semi-lines, say $\ell_1, \ldots, \ell_m$, each of which emanates from vertex B and crosses side $\Gamma\Delta$ of $\mathcal{R}$, so that the angle formed by $B\Gamma$ and semi-line ℓ_i is equal to $(i-1) \cdot \widehat{\Delta B\Gamma}/(m-1)$ for each $i = 1, \ldots, m$. These semi-lines split angle $\widehat{\Delta B\Gamma}$ into $m-1$ angles, each of which is equal to $\pi/4 \cdot (m-1)$, since $\widehat{\Delta B\Gamma} = \pi/4$. Say $\phi = \pi/4 \cdot (m-1)$. Then, we place vertices u^1_i at the intersection of semi-line l_i and $\Gamma\Delta$ for each $i = 1, \ldots, m$ (Fig. 1b). Symmetrically, we compute the position of the vertices of V_2 alongside AB of $\mathcal{R}$.

We denote by a_i the horizontal distance between two consecutive vertices u^1_i and u^1_{i+1}, $i = 1, \ldots, m-1$. We also define an additional bundle of m semi-lines, say $\ell'_1, \ldots, \ell'_m$, that emanate from vertex A. More precisely, semi-line l'_i emanates from vertex A and passes through the intersection of l_{m-i} and $\Gamma\Delta$ (i.e. vertex u^1_{m-i}) for each $i = 1, \ldots, m$ (Fig. 1b). Let ϕ'_i be the angle formed by two consecutive semi-lines l'_i and l'_{i+1} for each $i = 1, \ldots, m-1$. We are now ready to investigate some geometric properties of the proposed construction.

LEMMA 3.1. *For each $i = 1, 2, \ldots, m-1$, it holds that $a_{i-1} < a_i$.*

Proof. By induction. For the base of the induction, we have to show that $a_1 < a_2$. First observe that $a_1 = H \tan\phi$ and $a_1 + a_2 = H \tan 2\phi$. Therefore,

$$a_2 = H(\tan 2\phi - \tan\phi) \overset{(1)}{=} a_1 \cdot (1 + \tan 2\phi \cdot \tan\phi).$$

However, both $\tan\phi$ and $\tan 2\phi$ are greater than zero, which immediately implies that $a_1 < a_2$. For the induction hypothesis, we assume that $\forall k$, with $k < m-1$, it holds that $a_{k-1} < a_k$ and we prove that $a_k < a_{k+1}$. Obviously, $a_1 + \cdots + a_k = H \tan k\phi$. Based on Equation (1) and similarly to the base of the induction, we have:

(i) $a_{k+1} = H \tan\phi \cdot (1 + \tan(k+1)\phi \cdot \tan k\phi)$;
(ii) $a_k = H \tan\phi \cdot (1 + \tan(k-1)\phi \cdot \tan k\phi)$.

To complete the proof, observe that $(k+1)\phi > (k-1)\phi$. □

LEMMA 3.2. *For each $i = 2, \ldots, m-1$, it holds that $\phi'_{i-1} > \phi'_i$.*

Proof. The proof is by induction. For the base of the induction, we have to prove that $\phi'_1 > \phi'_2$ or equivalently

that $\tan\phi'_1 > \tan\phi'_2$. It holds that $\tan\phi'_1 = a_{m-1}/H$ and $\tan(\phi'_1+\phi'_2) = (a_{m-1}+a_{m-2})/H$. By combining these relationships with Equation (1), we have that $\tan\phi'_2 = Ha_{m-2}/(H^2+a_{m-1}^2+a_{m-1}a_{m-2})$. Therefore,

$$\tan\phi'_1 > \tan\phi'_2 \Leftrightarrow H^2(a_{m-1}-a_{m-2})+a_{m-1}^3+a_{m-1}^2a_{m-2}>0,$$

which trivially holds due to Lemma 3.1. For the induction hypothesis, we assume that $\forall k$, with $k < m-1$, it holds that $\phi'_{k-1} > \phi'_k$ and we have to show that $\phi'_k > \phi'_{k+1}$. Observe that

$$\begin{aligned}\tan\phi'_k &= \frac{\tan(\phi'_1+\cdots+\phi'_k)-\tan(\phi'_1+\cdots+\phi'_{k-1})}{1+\tan(\phi'_1+\cdots+\phi'_k)\cdot\tan(\phi'_1+\cdots+\phi'_{k-1})}\\ &= \frac{Ha_{m-k}}{H^2+(a_{m-1}+\cdots+a_{m-k})(a_{m-1}+\cdots+a_{m-k+1})},\\ \tan(\phi'_{k+1}) &= \frac{Ha_{m-(k+1)}}{H^2+(a_{m-1}+\cdots+a_{m-(k+1)})(a_{m-1}+\cdots+a_{m-k})}.\end{aligned}$$

By Lemma 3.1 we have that $(a_{m-1}+\cdots+a_{m-(k-1)}) < (a_{m-1}+\cdots+a_{m-(k+1)})$ and $H\cdot a_{m-k} > H\cdot a_{m-(k+1)}$. Therefore, $\tan\phi'_k > \tan\phi'_{k+1}$. □

LEMMA 3.3. *It holds that $\phi'_{m-1} \le \phi$.*

Proof. We equivalently prove that $\tan\phi'_{m-1} \le \tan\phi$.

$$\begin{aligned}\tan\phi'_{m-1} \le \frac{a_1}{H} &\Leftrightarrow \frac{a_1/H}{1+((a_1+\cdots+a_{m-1})/H)\cdot((a_2+\cdots+a_{m-1})/H)} \le \frac{a_1}{H}\\ &\Leftrightarrow \frac{1}{1+(H-a_1)/H} \le 1\\ &\Leftrightarrow a_1 \le H,\end{aligned}$$

which obviously holds. □

LEMMA 3.4. *Angle ϕ'_{m-1} is the smallest angle among all the angles formed in the drawing.*

Proof. From Lemma 3.2, it follows that angle ϕ'_{m-1} is the smallest angle among all ϕ'_i, $i = 1,\ldots,m-1$. Additionally, each angle ϕ'_i gets larger when the endpoint of the bundle (i.e. vertex u_n^2) moves alongside AB toward the intersection point of AB and the perpendicular bisector of line segment $\overline{u^1_{m-(i-1)}u^1_{m-i}}$, where it obtains its greatest value. From that point on and to its right, it continuously decreases, until it gets its minimum value when vertex u_1^2 is reached. At this point, it becomes equal to ϕ, which by Lemma 3.3 is greater than angle ϕ'_{m-1}. Therefore, ϕ'_{m-1} is the smallest angle among all the angles formed by pairs of consecutive edges incident on any vertex of V_2. Since $m \ge n$, the same holds for the vertices of V_1. Therefore, ϕ'_{m-1} defines the angular resolution of the drawing.

We now proceed to show that the crossing resolution of the constructed drawing is always greater than the angular resolution. To realize this, consider two crossing edges (refer to the bold, crossing dashed-edges of Fig. 1b). Their crossing defines (a) a pair of angles that are smaller than 90° and (b) another pair of angles that are larger than 90°. Obviously, only the acute angles participate in the computation of the crossing resolution (see angle ϕ_c in Fig. 1b). However, in a complete bipartite graph the acute angles formed by two crossing edges are always exterior to a triangle having two of its vertices on V_1 and V_2, respectively (refer to the gray-colored triangle of Fig. 1b). Therefore, the crossing resolution is always greater than the angular resolution, as desired. □

LEMMA 3.5. *It holds that $\phi'_{m-1} \ge \phi/2$.*

Proof. We equivalently prove that $\tan\phi'_{m-1} > \tan(\phi/2)$. Let x be the length of edge (u_1^2, u_2^1) in the produced drawing. Then, by Equation (2) and since $x > H$, we have the following:

$$\tan\frac{\phi}{2} = \frac{a_1/x}{1+H/x} < \frac{a_1}{2H}.$$

From the above relationship, it follows that in order to complete the proof of this lemma, it is enough to prove that $\tan\phi'_{m-1} > a_1/2H$. So, we have

$$\begin{aligned}\tan\phi'_{m-1} > \frac{a_1}{2H} &\Leftrightarrow \frac{a_1/H}{1+((a_1+\cdots+a_{m-1})/H)\cdot((a_2+\cdots+a_{m-1})/H)} > \frac{a_1}{2H}\\ &\Leftrightarrow \frac{1}{1+(H-a_1)/H} > \frac{1}{2}\\ &\Leftrightarrow a_1 > 0,\end{aligned}$$

which obviously holds. □

THEOREM 3.2. *A complete bipartite graph $K_{m,n}$ admits a two-layered drawing of total resolution $O(1/\max\{m,n\})$, which is asymptotically optimal.*

Proof. The proof id immediately follows from Lemmas 3.4 and 3.5. □

Assume, without loss of generality, that $\mathcal{L}_1$ and $\mathcal{L}_2$ are two horizontal lines, $\mathcal{L}_2$ coincides with x-axis and the drawing produced by our algorithm has $a_1 = 1$. Then, we can express the height of drawing $\Gamma(K_{m,n})$ as a function of ϕ, as follows:

$$a_1 = 1 \Longleftrightarrow \tan\phi\cdot H = 1 \Longleftrightarrow H = 1/\tan\phi.$$

Based on the above, the area of $\Gamma(K_{m,n})$, which is equal to H^2, is bounded by $1/\tan^2\phi$. By Equation (3), this is further bounded by $1/\phi^2$. By Theorem 3.2, it holds that $\phi = O(1/\max\{m,n\})$. Therefore, the total area occupied by the drawing is $O(\max\{m^2,n^2\})$. However, the vertices of $K_{m,n}$ in

$\Gamma(K_{m,n})$ are not necessarily located at grid points. To achieve this, we move the horizontal line $\mathcal{L}_1$ to the horizontal grid line immediately above it and each vertex of both V_1 and V_2 to the rightmost grid-point to its left. In this manner, we obtain a new drawing $\Gamma'(K_{m,n})$, in which every vertex is located at a grid point. By Lemma 3.1, it follows that there are no two vertices sharing the same gridpoint, since a_1 is equal to one grid unit. Since neither horizontal line $\mathcal{L}_1$ nor any vertex of $K_{m,n}$ moves more than one unit of length, the total resolution of $\Gamma'(K_{m,n})$ is not asymptotically affected, and, in addition, the height of the drawing is not significantly greater (i.e. asymptotically it remains the same). The following theorem summarizes this result.

THEOREM 3.3. *A complete bipartite graph $K_{m,n}$ admits a two-layered grid drawing of a total resolution of $\Theta(1/\max\{m,n\})$ and area of $O(\max\{m^2, n^2\})$.*

4. A FORCE-DIRECTED ALGORITHM

We present a force-directed algorithm that, starting from an initial drawing computed by a classical force-directed technique, results in a drawing of improved total resolution. The algorithm reinforces the classical force-directed algorithm of Eades [27] with some additional forces exerted to the vertices of the graph. More precisely, these additional forces involve springs and some extra attractive or repulsive forces on vertices with degree greater than 1 and on the endpoints of edges that are involved in an edge crossing. This aims to ensure that the angles between incident edges and the angles formed by pairs of crossing edges will be as large as possible in an equilibrium state of the model.

The classical force-directed algorithm of Eades [27] models the vertices of the graph as electrically charged particles that repel each other, and its edges by springs in order to attract adjacent vertices. Based on experimental results, we employ the attractive forces of the classical force-directed algorithm of Eades and omit the repulsive ones. In addition, we have chosen to use springs that follow the logarithmic law instead of the Hooke's law, in order to avoid exerting strong forces on distant vertices. More precisely, the attractive forces (denoted by $\mathcal{F}_{\texttt{spring}}$) follow the following formula:

$$\mathcal{F}_{\texttt{spring}}(p_u, p_v) = C_{\texttt{spring}} \cdot \log \frac{||p_u - p_v||}{\ell_{\texttt{spring}}} \cdot \overrightarrow{p_u p_v}, \quad (u, v) \in E,$$

where $C_{\texttt{spring}}$ and $\ell_{\texttt{spring}}$ capture the *stiffness* and the *natural length* of the springs, respectively. Recall that $\overrightarrow{p_u p_v}$ denotes the unit length vector from p_u to p_v.

We first describe our approach for the case where two edges, say $e = (u, v)$ and $e' = (u', v')$, are involved in a crossing. Let p_c be their intersection point (see the gray-colored point of Fig. 2). Without loss of generality, we assume that u is to the left of v, u' to the left of v', $y_{u'} < y_u$ and $y_v < y_{v'}$, as in Fig. 2. Let $\theta_{vv'}$ be the angle formed by the line segments $\overline{p_c p_v}$ and $\overline{p_c p_{v'}}$ in a counter-clockwise order around u from $\overline{p_c p_v}$ to $\overline{p_c p_{v'}}$. To avoid confusion, we assume that $\theta_{vv'} = \theta_{v'v}$, i.e. we abuse the counter-clockwise measurement of the angles that would result in $\theta_{vv'} = 2\pi - \theta_{v'v}$. Similarly, we define the remaining angles of Fig. 2. Obviously, $\theta_{vv'} + \theta_{v'u} = \pi$. Ideally, we would like $\theta_{vv'} = \theta_{v'u} = \pi/2$, i.e. e and e' form an RAC. As we will shortly see, the magnitude of the forces that we apply on the vertices u, u', v and v' depends on (a) the angles $\theta_{vv'}$ and $\theta_{v'u}$ and (b) the lengths of the line segments $\overline{p_c p_u}$, $\overline{p_c p_{u'}}$, $\overline{p_c p_v}$ and $\overline{p_c p_{v'}}$.

The physical model that describes our approach is illustrated in Fig. 2. Initially, for each pair of crossing edges at point p_c, we place springs connecting consecutive vertices in the counter-clockwise order around p_c, as in Fig. 2a. The magnitude of the forces due to these springs should capture our preference for right angles. Consider the spring connecting v and v'. The remaining ones are treated symmetrically. We set the natural length, say $\ell^{vv'}_{\texttt{spring}}$, of the spring connecting the vertices v and v' to be $\sqrt{\|p_c - p_v\|^2 + \|p_c - p'_v\|^2}$. This quantity corresponds to the length of the line segment that connects v and v' in the

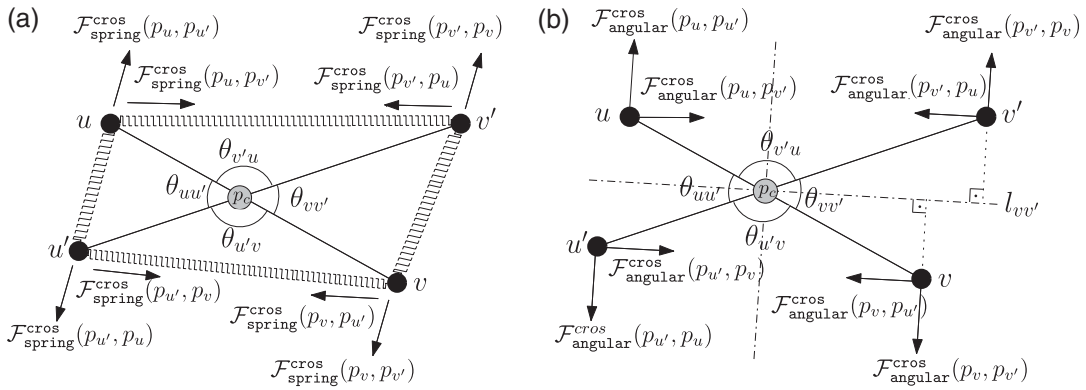

FIGURE 2. Forces applied on vertices in order to maximize the crossing resolution. (**a**) Springs on vertices involved in crossing. (**b**) Repelling or attractive forces based on the angles.

optimal case where $\theta_{vv'} = \pi/2$. So, in an equilibrium state of this model on a graph consisting only of e and e', edges e and e' will form an RAC. In conclusion, the force on v due to the spring of v' is defined as follows:

$$\mathcal{F}^{\texttt{cros}}_{\texttt{spring}}(p_v, p_{v'}) = C^{\texttt{cros}}_{\texttt{spring}} \cdot \log \frac{\|p_v - p_{v'}\|}{\ell^{vv'}_{\texttt{spring}}} \cdot \overrightarrow{p_v p_{v'}}.$$

The remaining forces of Fig. 2a are defined similarly. Note that, in the formula above, the constant $C^{\texttt{cros}}_{\texttt{spring}}$ is used to control the stiffness of the springs.

So far, we have managed to express our preference for RACs based on the lengths of the line segments $\overline{p_c p_u}$, $\overline{p_c p_{u'}}$, $\overline{p_c p_v}$ and $\overline{p_c p_{v'}}$. The same can be achieved using the angles $\theta_{vv'}$ and $\theta_{v'u}$ (Fig. 2b). We again restrict our description on the angle formed by the line segments $\overline{p_c p_v}$ and $\overline{p_c p_{v'}}$. Ideally, we would like to exert forces on the vertices v and v' such that: (i) when $\theta_{vv'} \to 0$, the magnitude of the force is very large (in order to repel v and v') and (ii) when $\theta_{vv'} \to \pi/2$, the magnitude of the force is very small. A function, say $f : \mathbb{R} \to \mathbb{R}$, which captures this property is: $f(\theta) = |\pi/2 - \theta|/\theta$. Having specified the magnitude of the forces, it remains to specify its direction. More precisely, we set the direction of the force on v (due to v') to be perpendicular to the line that bisects the angle $\theta_{vv'}$ (refer to the dash-dotted line $l_{vv'}$ of Fig. 2b), or equivalently parallel to the unit length vector $\texttt{Perp}(\texttt{Bsc}(\overrightarrow{p_c p_v}, \overrightarrow{p_c p_{v'}}))$. Recall that `Perp` and `Bsc` refer to the perpendicular and bisector vectors, respectively (Section 2). It is clear that if $\theta_{vv'} < \pi/2$, the forces on v and v' should be repulsive (in order to enlarge the angle between them), otherwise attractive. This can be captured by the *sign* function. Summarizing the above, we use the following formula, which expresses the force on v due to v'.

$$\mathcal{F}^{\texttt{cros}}_{\texttt{angle}}(p_v, p_{v'}) = C^{\texttt{cros}}_{\texttt{angle}} \cdot \text{sign}\left(\theta_{vv'} - \frac{\pi}{2}\right) \cdot f(\theta_{vv'}) \cdot \texttt{Perp}(\texttt{Bsc}(\overrightarrow{p_c p_v}, \overrightarrow{p_c p_{v'}})),$$

where constant $C^{\texttt{cros}}_{\texttt{angle}}$ controls the strength of the force. Similarly, we define the remaining forces of Fig. 2b.

Consider now a vertex $u \in V$ that is incident on $d(u)$ edges, say $e_0 = (u, v_0), e_1 = (u, v_1), \ldots, e_{d(u)-1} = (u, v_{d(u)-1})$; see Fig. 3a. We assume that $e_0, e_1, \ldots, e_{d(u)-1}$ are consecutive in the counter-clockwise order around u in the drawing of the graph. Similarly to the case of two crossing edges, we proceed to connect the endpoints of consecutive edges around u by springs, as in Fig. 3a. In this case, the natural length of each spring, should capture our preference for angles equal to $2\pi/d(u)$ (i.e. the obvious upper bound). To achieve this, we proceed as follows: For each $i = 0, 1, \ldots, d(u) - 1$, we set the natural length, say $l^i_{\texttt{spring}}$, of the spring connecting v_i with $v_{(i+1) \bmod (d(u))}$, to be

$$\ell^i_{\texttt{spring}} = \sqrt{\|a_i\|^2 + \|b_i\|^2 - 2\|a_i\|\|b_i\| \cdot \cos(2\pi/d(u))},$$

where $a_i = e_i$ and $b_i = e_{(i+1) \bmod (d(u))}$. The quantity $\ell^i_{\texttt{spring}}$ corresponds to the length of the line segment that connects v_i with $v_{(i+1) \bmod (d(u))}$ in the optimal case where the angle formed by e_i and $e_{(i+1) \bmod (d(u))}$ is $2\pi/d(u)$ for each $i = 0, 1, \ldots, d(u) - 1$. Therefore, the spring forces between consecutive edges follow the formula

$$\mathcal{F}^{\texttt{angular}}_{\texttt{spring}}(p_{v_i}, p_{v_{(i+1) \bmod (d(u))}}; u) = C^{\texttt{angular}}_{\texttt{spring}} \cdot \log \frac{\|p_{v_i} - p_{v_{(i+1) \bmod (d(u))}}\|}{\ell^i_{\texttt{spring}}} \cdot \overrightarrow{p_{v_i} p_{v_{(i+1) \bmod (d(u))}}},$$

where the quantity $C^{\texttt{angular}}_{\texttt{spring}}$ is a constant that captures the stiffness of the spring.

Now let θ_i be the angle formed by e_i and $e_{(i+1) \bmod (d(u))}$, measured in the counter-clockwise direction from e_i to $e_{(i+1) \bmod (d(u))}$, $i = 0, 1, \ldots, d(u) - 1$. Similarly to the case of two crossing edges, we exert forces on v_i and $v_{(i+1) \bmod (d(u))}$ perpendicular to the bisector of θ_i, as illustrated in Fig. 3b. However, in this case, we need a magnitude function such that: (i) when $\theta_i \to 0$, the magnitude of the force is very large (in order to repel v_i and $v_{(i+1) \bmod (d(u))}$) and (ii) when $\theta_i \to 2\pi/d(u)$, the magnitude of the force is very small. Such

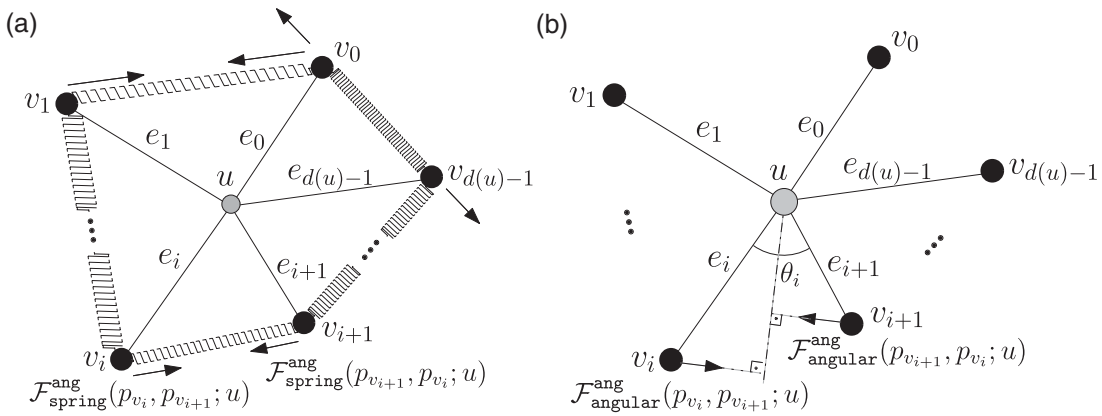

FIGURE 3. Forces applied on vertices in order to maximize the angular resolution. (**a**) Springs on consecutive edges around u. (**b**) Repelling or attractive forces based on the angles.

a function, say $g : \mathbb{R} \times V \to \mathbb{R}$, is $g(\theta; u) = |2\pi/d(u) - \theta|/\theta$. Having fully specified the forces applied on the endpoints of consecutive edges and their directions, we are now ready to provide the exact formulas that the forces follow:

$$\mathcal{F}^{\texttt{angular}}_{\texttt{angle}}(p_{v_i}, p_{v_{(i+1) \bmod (d(u))}}; u) = C^{\texttt{angular}}_{\texttt{angle}} \cdot \text{sign}\left(\theta_i - \frac{2\pi}{d(u)}\right) \cdot g(\theta_i; u) \cdot \texttt{Perp}(\texttt{Bsc}(\overrightarrow{p_u p_{v_i}}, \overrightarrow{p_u p_{v_{(i+1) \bmod (d(u))}}})),$$

where $C^{\texttt{angular}}_{\texttt{angle}}$ is a constant to control the strength of the force. We note that our experimental evaluation has shown that forces $\mathcal{F}^{\texttt{x}}_{\texttt{cros}}$ and $\mathcal{F}^{\texttt{x}}_{\texttt{angle}}$, where $\texttt{x} \in \{\texttt{cros}, \texttt{angular}\}$ have complementary effect and produce better drawings when they are both present. We stimulate that this is due to the fact that the movement of the vertices is determined as a result of a larger number of vectors. In addition, by setting zero values to either $C^{\texttt{cros}}_{\texttt{spring}}$ and $C^{\texttt{cros}}_{\texttt{angle}}$, or, $C^{\texttt{angular}}_{\texttt{spring}}$ and $C^{\texttt{angular}}_{\texttt{angle}}$, our algorithm can be configured to maximize only the angular or the crossing resolution, respectively.

4.1. Time complexity analysis

Our approach is outlined in Algorithm 1. On each iteration, the algorithm computes three types of forces. Type-1 forces correspond to attractive forces of the classical force-directed model among pairs of adjacent vertices of the graph. Their computation requires $O(E)$ time per iteration. In Type-2 forces of Algorithm 1, we compute forces due to edge crossings. In a straightforward manner, the computation of all edges that are involved in crossings in Line 7 of Algorithm 1 needs $O(E^2)$ time. Having computed all pairs of crossings edges, we can compute their associated forces in constant time for each pair of crossing edges. Therefore, Type-2 forces can be computed in $O(E^2)$ time per iteration. In Type-3 forces of Algorithm 1, we compute forces due to the angles between consecutive edges. To cope with this case, initially we have to sort the incident edges of each vertex of the graph in cyclic order (see Line 16 of Algorithm 1). This can be done in $O(d(G) \log d(G))$ time, where $d(G)$ denotes the degree of the graph. Lines 18–20, where we compute the forces, need an extra $O(d(G))$ time. Thus, Type-3 forces needs $O(E + Vd(G) \log d(G))$ time per iteration. Summarizing the above, each iteration of Algorithm 1 takes $O(E^2 + V(V + d(G) \log d(G)))$ time.

The computational complexity of Algorithm 1 can be further improved using standard techniques adopted from computational geometry [39–41]. More precisely, if K is the number of pairwise-crossing edges, then the K intersections in Type-2 force can be reported in $O(K + E \log^2 E / \log \log E)$ time [41, p. 277], which leads to a total time complexity $O(K + E \log^2 E / \log \log E + V(V + d(G) \log d(G)))$ per iteration.

Algorithm 1: Force-directed algorithm.

Input : An undirected graph G and an initial placement $P = (p_v)_{v \in V}$.
Output: A drawing of G of large angular and crossing resolution.

1 **for** ($t \leftarrow 1$ **to** ITERATIONS) **do**
2 {Type 1: Spring forces.}
3 **foreach** $u \in V$ **do**
4 $\mathcal{F}_u(t) \leftarrow \sum_{v:(u,v) \in E} \mathcal{F}_{\texttt{spring}}(p_v, p_u)$;
5 **end**
6 {Type 2: Forces applied on vertices in order to maximize the crossing resolution.}
7 **foreach** (pair of intersecting edges $e = (u, v)$ and $e' = (u', v')$) **do**
8 // The relative positions of edges e and e' are illustrated in Fig. 2.;
9 $\mathcal{F}_v(t)$ += $\mathcal{F}^{\texttt{cros}}_{\texttt{spring}}(p_v, p_{v'}) + \mathcal{F}^{\texttt{cros}}_{\texttt{spring}}(p_v, p_{u'}) + \mathcal{F}^{\texttt{cros}}_{\texttt{angle}}(p_v, p_{v'}) + \mathcal{F}^{\texttt{cros}}_{\texttt{angle}}(p_v, p_{u'})$;
10 $\mathcal{F}_{v'}(t)$ += $\mathcal{F}^{\texttt{cros}}_{\texttt{spring}}(p_{v'}, p_u) + \mathcal{F}^{\texttt{cros}}_{\texttt{spring}}(p_{v'}, p_v) + \mathcal{F}^{\texttt{cros}}_{\texttt{angle}}(p_{v'}, p_u) + \mathcal{F}^{\texttt{cros}}_{\texttt{angle}}(p_{v'}, p_v)$;
11 $\mathcal{F}_u(t)$ += $\mathcal{F}^{\texttt{cros}}_{\texttt{spring}}(p_u, p_{v'}) + \mathcal{F}^{\texttt{cros}}_{\texttt{spring}}(p_u, p_{u'}) + \mathcal{F}^{\texttt{cros}}_{\texttt{angle}}(p_u, p_{v'}) + \mathcal{F}^{\texttt{cros}}_{\texttt{angle}}(p_u, p_{u'})$;
12 $\mathcal{F}_{u'}(t)$ += $\mathcal{F}^{\texttt{cros}}_{\texttt{spring}}(p_{u'}, p_v) + \mathcal{F}^{\texttt{cros}}_{\texttt{spring}}(p_{u'}, p_u) + \mathcal{F}^{\texttt{cros}}_{\texttt{angle}}(p_{u'}, p_v) + \mathcal{F}^{\texttt{cros}}_{\texttt{angle}}(p_{u'}, p_v)$;
13 **end**
14 {Type 3: Forces applied on vertices in order to maximize the angular resolution.}
15 **foreach** ($u \in V$ with $d(u) > 1$) **do**
16 $e_0, \ldots, e_{d(u)-1} \leftarrow$ incident edges of u in counter-clockwise order (see Fig. 3);
17 Let $e_i = (u, v_i), i = 0, \ldots d(u) - 1$;
18 **for** ($i \leftarrow 0$ **to** $d(u) - 1$) **do**
19 $\mathcal{F}_u(t)$ += $\mathcal{F}^{\texttt{angular}}_{\texttt{spring}}(p_{v_i}, p_{v_{(i+1) mod (d(u))}}; u) + \mathcal{F}^{\texttt{angular}}_{\texttt{angle}}(p_{v_i}, p_{v_{(i+1) mod (d(u))}}; u)$;
20 **end**
21 **end**
22 **foreach** $u \in V$ **do**
23 $p_u \leftarrow p_u + \delta \cdot \mathcal{F}_u(t)$;
24 **end**
25 **end**

4.2. Similarities and differences with previous techniques

In the work of Lin and Yen [32], the technique that applies large repelling forces perpendicular to the bisectors of the angles

formed by consecutive edges of a vertex, when these angles are small, is referred to as *edge–edge repulsion*. In their work, the function, say $g : \mathbb{R} \to \mathbb{R}$, which controls the magnitude of the force is $g(\theta) = \cot(\theta/2)$, where θ is the angle formed by two consecutive edges incident on a common vertex. Observe that g has the same property as the one we used, i.e. $g(\theta)$ is very large, when $\theta \to 0$. We preferred to use a different function to control the magnitude of the forces in order to maintain a uniform approach in both crossing and angular cases.

Huang *et al.* [36], who have independently published a similar to our force-directed algorithm, use only 'angular' forces (i.e. that depend only on the angles formed either by edge crossings or by consecutive edges incident on a common vertex) with different functions to control their magnitudes and different directions. They do not use 'spring' forces to affect the angular or the crossing resolution (apart from the ones of the classical force-directed algorithm of Eades [27]). In Section 4.3, we provide an experimental comparison of these techniques.

In Didimo *et al.* [33, 34], the crossing resolution is improved in a post-processing step in which every pair of crossing edges is associated with a disk centered at the crossing point. The intersection points of this disk with the edges participating at the crossing define four new vertices that form a four-cycle, called *cage*. Appropriate forces are then applied so that the cages are eventually drawn as close to squares as possible, which enforces their diagonals to cross at large angles. Since these techniques result in drawings with bends, they are not included in our experimental comparison.

4.3. Experimental results

In this section, we present the results of the experimental evaluation of our algorithm. Apart from our algorithm, we have implemented the classical force-directed algorithm of Eades [27], the algorithm of Lin and Yen [32] and the algorithm of Huang *et al.* [36]. The implementations are in Java using the yFiles library (http://www.yworks.com). The experiment was performed on a Linux machine with 2.00 GHz CPU and 2 GB RAM using the Rome graphs (a collection of around 11.500 graphs) obtained from graphdrawing.org. Figure 8 illustrates a drawing of a Rome graph with 99 vertices and 135 edges produced by our force-directed algorithm. Even though our force-directed algorithm focuses on angular, crossing and total resolution, we provide data on the performance of the algorithm with respect to the total area, edge length and number of crossings (Fig. 5).

The experiment was performed as follows. First, each Rome graph was laid out using the SmartOrganic layouter of yFiles. This layout was the input layout of all algorithms, in order to speed up the experiment and overcome problems associated with local minimum traps, especially in large graphs. If both the angular and the crossing resolution between two consecutive iterations of each algorithm were not improved more than 0.001°, we assumed that the algorithm has converged and we did not proceed any more. We note that the termination condition

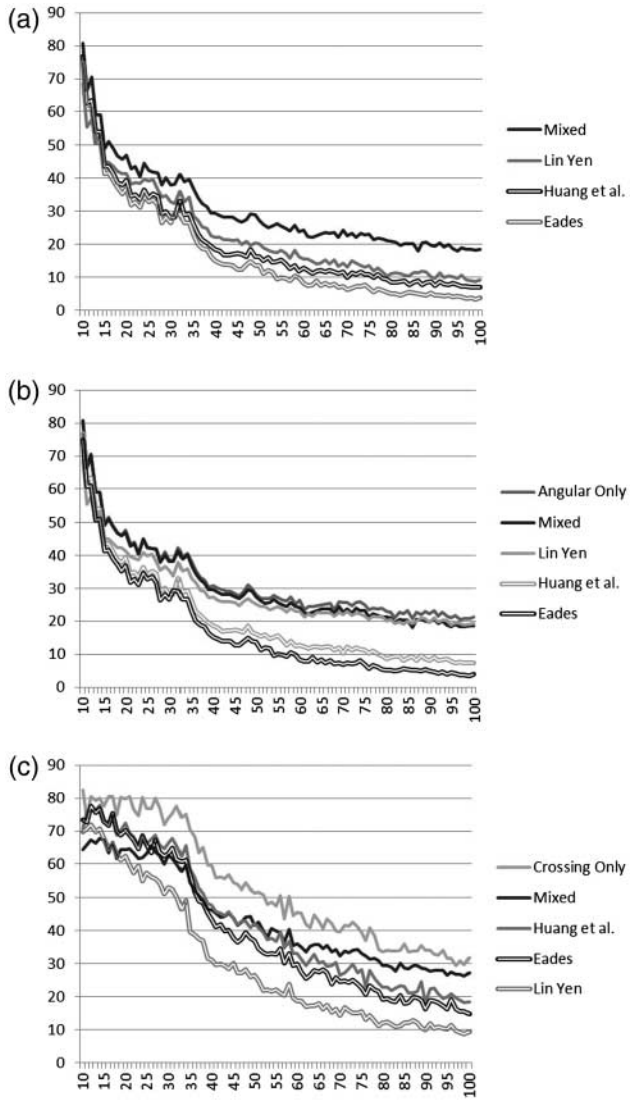

FIGURE 4. A visual presentation of our experimental results on *resolution*: the X-axis corresponds to the number of vertices of the graph and the Y-axis to the resolution measured in degrees. (**a**) Total resolution results. (**b**) Angular resolution results. (**c**) Crossing resolution results.

is quite strict and demands a large number of iterations. The maximum number of iterations that an algorithm could perform in order to converge was set to 100.000. However, very few graphs required the maximum number of iterations in order to converge to a layout. Once an algorithm converged, we measured its angular, crossing and total resolution, its average edge length, the number of crossings, running time and required number of iterations to converge. We note that our algorithm is evaluated as (a) Angular-Only, (b) Crossing-Only and (c) Mixed. The results are illustrated in Figs 4–6. The values plotted in Figs 4–6 are average values over all graphs in the Rome database of a specific size (i.e. number of vertices).

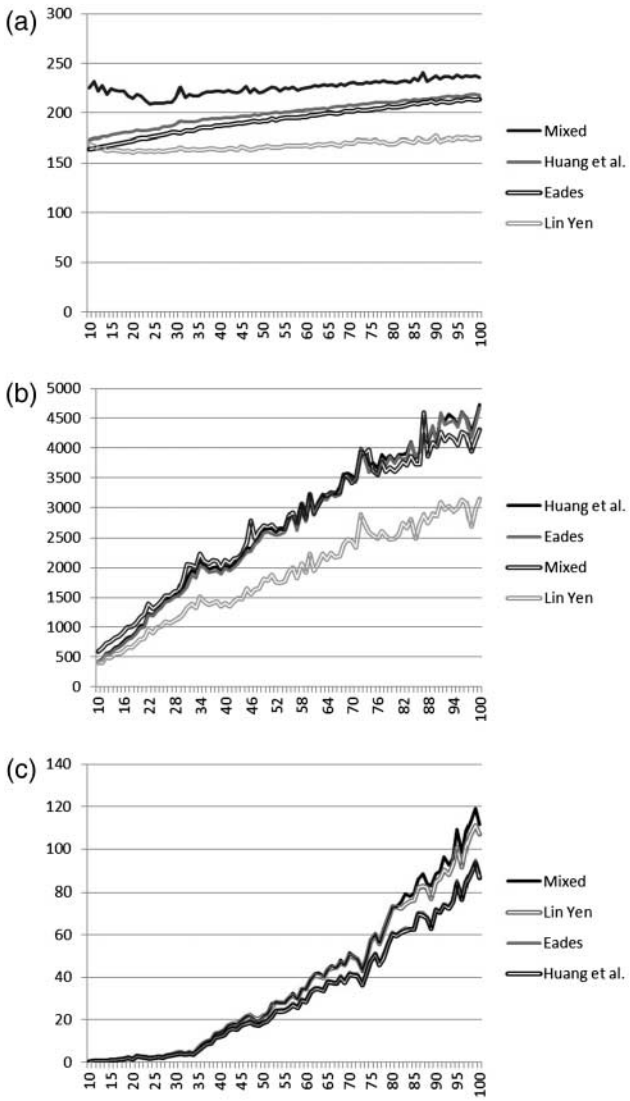

FIGURE 5. A visual presentation of our experimental results on *common graph drawing metrics:* The X-axis indicates the number of vertices of the graph. (**a** and **b**) The Y-axis corresponds to the area and the average edge length of the produced drawings, respectively. (**c**) The Y-axis corresponds to the number of crossings of the produced drawings.

The total resolution maximization problem: The plot in Fig. 4a indicates that our Mixed algorithm clearly outperforms all other drawing algorithms of the experiment. More specifically, in 80.46% (respectively, 90.83%) of the graphs of the experiment, the Mixed algorithm yields a better solution compared with the algorithm of Lin–Yen (respectively, Huang *et al.*) with an average improvement of 10.21° (respectively, 11.63°). The plot in Fig. 4a also shows that the Mixed algorithm managed to produce drawings of total resolution of ∼20° for graphs of more than 50 vertices in the Rome database. An example of such a drawing is given in Fig. 8.

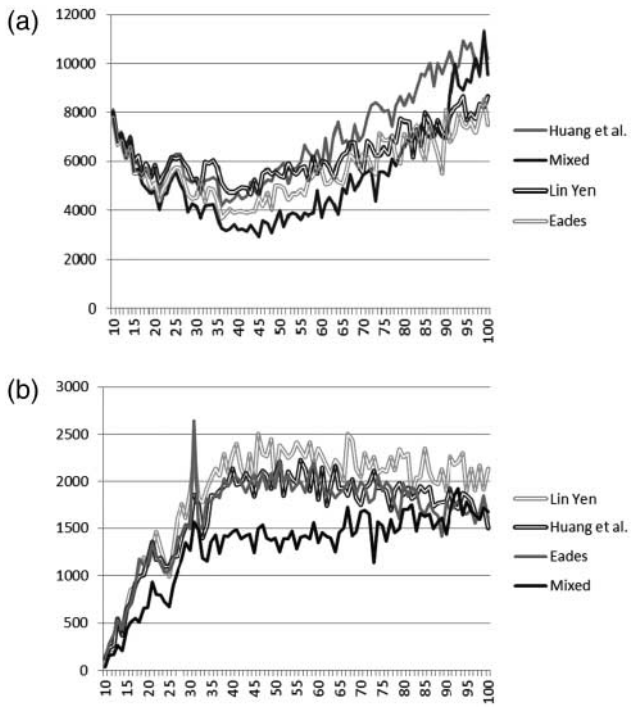

FIGURE 6. A visual presentation of our experimental results on *algorithm's efficiency:* The X-axis indicates the number of vertices of the graph. (**a** and **b**) The Y-axis corresponds to the running time measured in milliseconds and the iterations needed to converge, respectively.

The angular resolution maximization problem: The plot in Fig. 4b shows that the Angular-Only algorithm yields drawings with better angular resolution compared with the other drawing algorithms of the experiment. More specifically, in 67.82% (respectively, 92.12%) of the graphs of the experiment, the Angular-Only algorithm yields a better solution compared with the algorithm of Lin–Yen (respectively, Huang *et al.*) with an average improvement of 6.73° (respectively, 12.75°). The angular resolution of the drawings produced by the Mixed algorithm is almost equal to the ones produced by the algorithm of Lin and Yen. Note that the algorithm of Lin and Yen, in contrast to ours, does not modify the embedding of the initial layout (see [32] for more details), i.e. it needs a close-to-final starting layout and improves on it. This explains why the Mixed algorithm achieves almost the same performance, in terms of angular resolution, as the one of Lin and Yen. More precisely, in 59.23% of the graphs, the Mixed algorithm yields a better solution compared with Lin–Yen's algorithm, with an average improvement of 6.49°.

The crossing resolution maximization problem: In Fig. 4c, the data were filtered to depict only the results of non-planar drawings produced by the algorithms and avoid infinity values in the case of planar ones. Clearly, the Crossing-Only algorithm outperforms the other drawing algorithms of the experiment, yielding drawings with better crossing

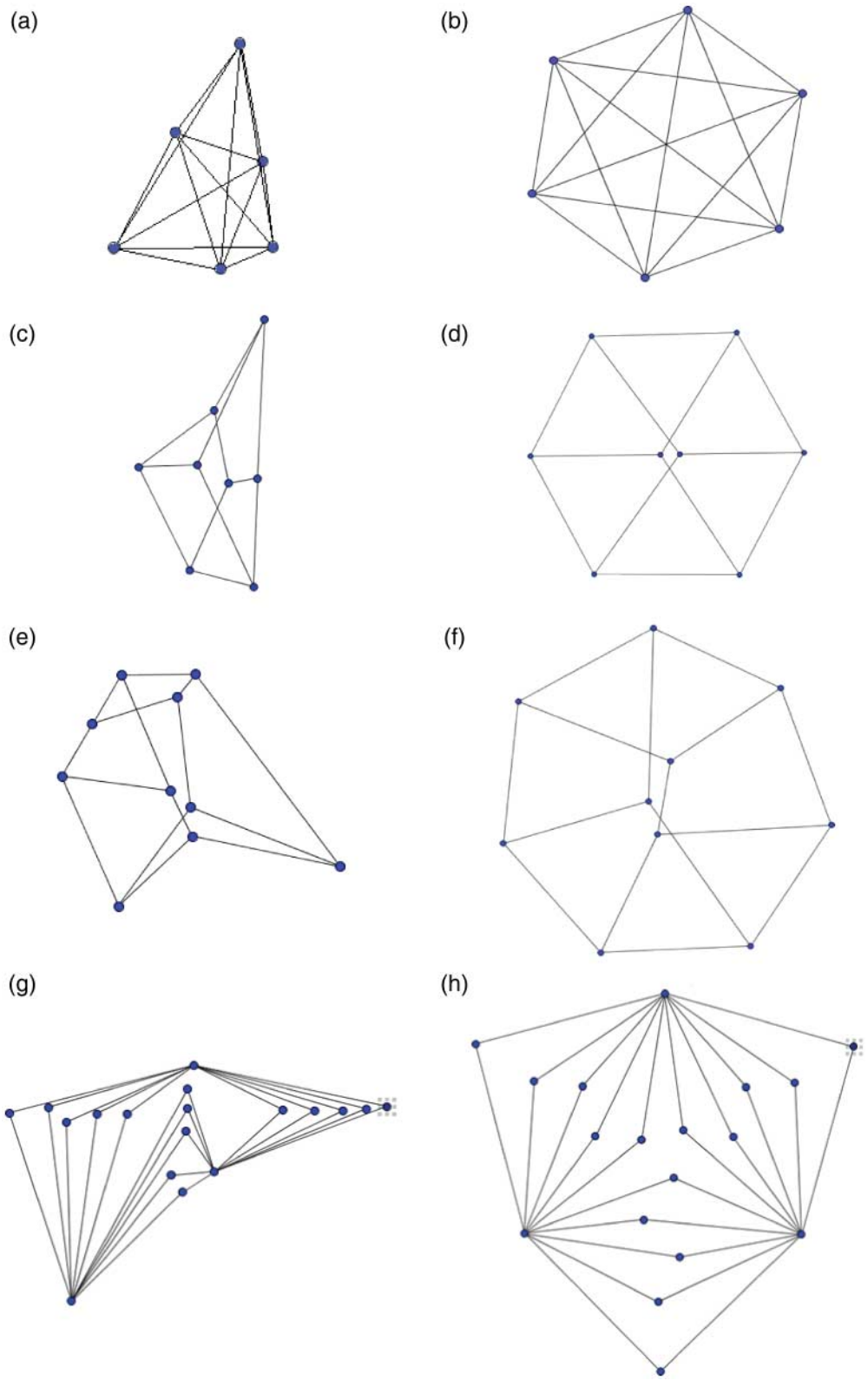

FIGURE 7. Sample drawings produced by our force-directed algorithm. The drawings on each line (of graphs) correspond to the same graph. The drawing on the left is the input of our algorithm, while the drawing on the right is its output. In the caption of each drawing (**a** and **b**) is translated as follows: (**a**) [(**b**), respectively] is the angular (crossing, respectively) resolution measured in degrees. Drawings of these graphs are also presented either in the classical work of Fruchterman and Reingold [28] or in the work of Lin and Yen [32].

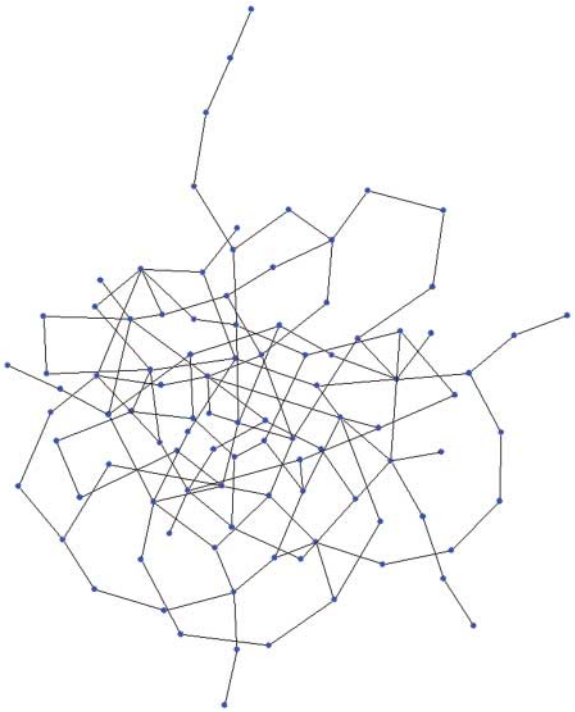

FIGURE 8. A drawing of Rome graph grafo10129.99 consisting of 99 vertices and 135 edges with angular resolution 20.15° and crossing resolution 26.12°.

resolution. More specifically, in 92.26% (respectively, 78.47%) of the graphs of the experiment, the Crossing-Only algorithm yields a better solution compared with the algorithm of Lin–Yen (respectively, Huang *et al.*). The average improvement implied by the Crossing-Only algorithm is 27.13° (respectively, 17.08°) w.r.t. the algorithm of Lin–Yen (respectively, Huang *et al.*). The Mixed algorithm yields a better solution in 85.23% (respectively, 54.62%) of the graphs of the experiment compared with the algorithm of Lin–Yen (respectively, Huang *et al.*) with an average improvement of 18.75° (respectively, 11.17°).

From the above discussion, it follows that our algorithm results in drawings with better resolution than the ones produced by the other algorithms of the experiment. However, the experimental evaluation also shows that the average edge length of the drawings produced by our algorithm is larger than the average edge length of the drawings produced by the other algorithms (Fig. 5a). The edge length is influenced by the natural spring length used by the force-directed algorithms. In the experiment, the natural length of the springs was set to 150 units. The average edge length of the drawings produced by the Mixed algorithm was 225.11 units, whereas the corresponding ones of Eades, Lin–Yen and Huang *et al.* were 191.61, 167.1 and 198.45 units, respectively. The experimental evaluation also shows that our algorithm produces drawings of comparable area to that required by the algorithms of Eades and Huang *et al.* All these algorithms are outperformed by the algorithm of Lin and Yen, which produced the most compact drawings. Despite the fact that the Mixed algorithm produces drawings of larger edge length compared with the drawings produced by the algorithms of Eades and Huang *et al.*, its produced drawings are of comparable area. This may be attributed to the fact that, as the experimental analysis shows, it produces drawings with a larger number of crossings. More precisely, Fig. 5c shows that the Mixed algorithm yields drawings of almost the same crossings as the one of Lin and Yen and slightly more than the algorithms of Eades and Huang *et al*.

Finally, Fig. 6a and b summarizes the running time performance of the algorithms and the iterations required to converge, respectively. The Mixed algorithm needs, on average, 5191 ms and 1298 iterations to converge, whereas the algorithm of Huang *et al.* (which is of the same time complexity) 6626 and 1694, respectively. The algorithm of Lin and Yen needs, on average, 6123 ms and 1920 iterations to converge. Note that the time complexity of Lin–Yen's algorithm is better than ours. However, the termination condition takes into account the crossing improvement and therefore the algorithm of Lin and Yen needs more iterations to converge, which explains this contradiction. In the running time experiment, we exclude the algorithm of Eades, since it does not take into account the angular or the crossing resolution and, therefore, it needs more iterations to converge, as depicted in Fig. 6b.

5. SAMPLE DRAWINGS

In this section, we present some sample drawings produced by our force-directed algorithm. Figure 7 depicts (some easy to produce) drawings of known graphs. Drawings of these graphs are also presented either in the classical work of Fruchterman and Reingold [28] or in the work of Lin and Yen [32]. In Fig. 8, we present a drawing of a Rome graph with 99 vertices and 135 edges.

6. CONCLUSIONS

In this paper, we introduced and studied the total resolution maximization problem. Our work leaves several open problems. It would be interesting to try to identify other classes of graphs that admit optimal drawings. Even the case of planar graphs is of interest, as by allowing some edges to cross (say at large angles), we may improve the angular resolution and therefore the total resolution. In [22], a class of graphs satisfying this property is given.

ACKNOWLEDGEMENTS

The authors would also like to thank the anonymous reviewers for their valuable comments and suggestions to improve the quality of the paper.

FUNDING

The work of E.N.A. has been co-financed by the European Union (European Social Fund—ESF) and Greek national funds through the Operational Program 'Education and Lifelong Learning' of the National Strategic Reference Framework (NSRF)—Research Funding Program: Heracleitus II. Investing in knowledge society through the European Social Fund.

REFERENCES

[1] Di Battista, G., Eades, P., Tamassia, R. and Tollis, I.G. (1994) Algorithms for drawing graphs: an annotated bibliography. *Comput. Geom.*, **4**, 235–282.

[2] Kaufmann, M. and Wagner, D. (eds) (2001) *Drawing Graphs: Methods and Models*. Lecture Notes in Computer Science 2025. Springer.

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

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

[5] Purchase, H.C., Carrington, D.A. and Allder, J.-A. (2002) Empirical evaluation of aesthetics-based graph layout. *Empir. Softw. Eng.*, **7**, 233–255.

[6] Ware, C., Purchase, H., Colpoys, L. and McGill, L. (2002) Cognitive measurements of graph aesthetics. *Inf. Vis.*, **1**, 103–110.

[7] Garey, M. and Johnson, D. (1983) Crossing number is NP-complete. *SIAM J. Algebr. Discret. Methods*, **4**, 312–316.

[8] Huang, W. (2007) Using Eye Tracking to Investigate Graph Layout Effects. *Proc. 6th Int. Asia-Pacific Symp. Visualization (APVIS 2007)*, Sydney, Australia, February 5–7, pp. 97–100. IEEE.

[9] Huang, W., Hong, S.-H. and Eades, P. (2008) Effects of Crossing Angles. *Proc. VGTC Pacific Visualization Symp. (PacificVis 2008)*, Kyoto, Japan, March 5–7, 2008 pp. 41–46. IEEE.

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

[11] Malitz, S.M. and Papakostas, A. (1994) On the angular resolution of planar graphs. *SIAM J. Discret. Math.*, **7**, 172–183.

[12] Garg, A. and Tamassia, R. (1994) Planar Drawings and Angular Resolution: Algorithms and Bounds. *Proc. 2nd Annual European Symp. (ESA '94)*, Utrecht, The Netherlands, September 26–28, Lecture Notes in Computer Science, pp. 12–23. Springer.

[13] Gutwenger, C. and Mutzel, P. (1999) Planar Polyline Drawings with Good Angular Resolution. *Proc. 6th Int. Symp. Graph Drawing (GD 1998)*, Montréal, Canada, August 13–15, Lecture Notes in Computer Science 1547, pp. 167–182. Springer.

[14] Bodlaender, H.L. and Tel, G. (2004) A note on rectilinearity and angular resolution. *J. Graph Algorithms Appl.*, **8**, 89–94.

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

[16] Arikushi, K., Fulek, R., Keszegh, B., Moric, F. and Toth, C. (2010) Graphs that Admit Right Angle Crossing Drawings. *Proc. 36th Int. Workshop on Graph Theoretic Concepts in Computer Science (WG 2010)*, Zarós, Crete, Greece, June 28–30, Lecture Notes in Computer Science, pp. 135–146. Springer.

[17] Angelini, P., Cittadini, L., Di Battista, G., Didimo, W., Frati, F., Kaufmann, M. and Symvonis, A. (2010) On the Perspectives Opened by Right Angle Crossing Drawings. *Proc. 17th Int. Symp. Graph Drawing (GD 2009)*, Chicago, IL, USA, September 22–25, Lecture Notes in Computer Science, pp. 21–32. Springer.

[18] Argyriou, E.N., Bekos, M.A. and Symvonis, A. (2011) The Straight-Line RAC Drawing Problem is NP-Hard. *Proc. 37th Int. Conf. Current Trends in Theory and Practice of Computer Science (SOFSEM 2011)*, Novy Smokovec, Slovakia, January 22–28, Lecture Notes in Computer Science, pp. 74–85. Springer.

[19] Di Giacomo, E., Didimo, W., Liotta, G. and Meijer, H. (2010) Area, Curve Complexity, and Crossing Resolution of Non-planar Graph Drawings. *Proc. 17th Int. Symp. Graph Drawing (GD 2009)*, Chicago, IL, USA, September 22–25, Lecture Notes in Computer Science, pp. 15–20. Springer.

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

[21] Di Giacomo, E., Didimo, W., Eades, P., Hong, S.-H. and Liotta, G. (2012) Bounds on the crossing resolution of complete geometric graphs. *Discrete Appl. Math.*, **160**, 132–139.

[22] van Kreveld, M. (2011) The Quality Ratio of RAC Drawings and Planar Drawings of Planar Graphs. *Proc. 18th Int. Symp. Graph Drawing (GD 2010)*, Konstanz, Germany, September 21–24, Lecture Notes in Computer Science, pp. 371–376. Springer.

[23] Eades, P. and Liotta, G. (2012) Right Angle Crossing Graphs and 1-Planarity. *Proc. 19th Int. Symp. Graph Drawing (GD 2011)*, Eindhoven, The Netherlands, September 21–23, Eindhoven, The Netherlands, September 21–23, Lecture Notes in Computer Science 7034, pp. 148–153. Springer.

[24] Dujmović, V., Gudmundsson, J., Morin, P. and Wolle, T. (2010) Notes on Large Angle Crossing Graphs. *Proc. 16th Symp. Computing: The Australasian Theory, CATS '10*, Brisbane, Australia, January 18–21, pp. 19–24. Australian Computer Society, Inc.

[25] Angelini, P., Di Battista, G., Didimo, W., Frati, F., Hong, S.-H., Kaufmann, M., Liotta, G. and Lubiw, A. (2011) Large Angle Crossing Drawings of Planar Graphs in Subquadratic Area. *Proc. XIV Spanish Meeting on Computational Geometry (EGC 2011)*, Alcalá de Henares, June 27–30, Encuentros de Geometría Computacional. pp. 125–128.

[26] Davidson, R. and Harel, D. (1996) Drawing graphs nicely using simulated annealing. *ACM Trans. Graph.*, **15**, 301–331.

[27] Eades, P. (1984) A heuristic for graph drawing. *Cong. Numer.*, **42**, 149–160.

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

[29] Kamada, T. and Kawai, S. (1989) An algorithm for drawing general undirected graphs. *Inf. Process. Lett.*, **31**, 7–15.

[30] Brandenburg, F., Himsolt, M. and Rohrer, C. (1996) An Experimental Comparison of Force-Directed and Randomized Graph Drawing Algorithms. *Proc. 3rd Int. Symp. Graph Drawing (GD 1995)*, Passau, Germany, September 20–22, Lecture Notes in Computer Science 1027, pp. 76–87. Springer.

[31] Bertault, F. (2000) A force-directed algorithm that preserves edge-crossing properties. *Inf. Process. Lett.*, **74**, 7–13.

[32] Lin, C.-C. and Yen, H.-C. (2005) A New Force-Directed Graph Drawing Method Based on Edge-Edge Repulsion. *Proc. 9th Int. Conf. Information Visualisation (IV 2005)*, London, UK, July 6–8, pp. 329–334. IEEE Computer Society.

[33] Didimo, W., Liotta, G. and Romeo, S. (2011) Topology-Driven Force-Directed Algorithms. In Brandes, U. and Cornelsen, S. (eds), *Proc. 18th Int. Symp. Graph Drawing (GD 2010)*, Lecture Notes in Computer Science 6502, Konstanz, Germany, September 21–24, pp. 165–176. Springer.

[34] Didimo, W., Liotta, G. and Romeo, S. (2011) A graph drawing application to web site traffic analysis. *J. Graph Algorithms Appl.*, **15**, 229–251.

[35] Argyriou, E.N., Bekos, M.A. and Symvonis, A. (2011) Maximizing the Total Resolution of Graphs. *Proc. 18th Int. Symp. Graph Drawing (GD 2010)*, Konstanz, Germany, September 21–24, Lecture Notes in Computer Science, pp. 62–67. Springer.

[36] Huang, W., Eades, P., Hong, S.-H. and Lin, C.-C. (2010) Improving Force-Directed Graph Drawings by Making Compromises Between Aesthetics. *Proc. IEEE Symp. Visual Languages and Human-Centric Computing (VL/HCC 2010)*, Leganés-Madrid, Spain, September 21–25, Lecture Notes in Computer Science, pp. 176–183. IEEE.

[37] Bárány, I. and Tokushige, N. (2004) The minimum area of convex lattice n-gons. *Combinatorica*, **24**, 171–185.

[38] Cheng, C.C., Duncanyz, C.A., Goodrichz, M.T. and Kobourovz, S.G. (1999) Drawing planar graphs with circular arcs. *Discrete Comput. Geom.*, **25**, 117–126.

[39] de Berg, M., van Kreveld, M., Overmars, M. and Schwarzkopf, O. (2000) *Computational Geometry: Algorithms and Applications* (2nd edn). Springer.

[40] O'Rourke, J. (1998) *Computational geometry in C* (2nd edn). Cambridge University Press.

[41] Preparata, F.P. and Shamos, M.I. (1985) *Computational Geometry: An Introduction*. Springer.