Advances in the Theory and Practice of Graph Drawing*

Roberto Tamassia
Department of Computer Science
Brown University
Providence, RI 02912–1910
rt@cs.brown.edu

Abstract

The visualization of conceptual structures is a key component of support tools for complex applications in science and engineering. Foremost among the visual representations used are drawings of graphs and ordered sets. In this talk, we survey recent advances in the theory and practice of graph drawing. Specific topics include bounds and tradeoffs for drawing properties, three-dimensional representations, methods for constraint satisfaction, and experimental studies.

1 Introduction

In this paper, we survey selected research trends in graph drawing, and overview some recent results of the author and his collaborators.

Graph drawing addresses the problem of constructing geometric representations of graphs, a key component of support tools for complex applications in science and engineering. Graph drawing is a young research field that has growth very rapidly in the last decade. One of its distinctive characteristics is to have furthered collaborative efforts between computer scientists, mathematicians, and applied researchers.

A comprehensive bibliography on graph drawing algorithms [21] cites more than 300 papers written before 1993. Most papers on graph drawing are cited in `geom.bib`, the computational geometry BibTEX bibliography available from `ftp://cs.usask.ca/pub/geometry/` (search for keyword "graph drawing"). Surveys on various aspects of graph drawing appear in [24, 32, 41, 44, 74, 75, 78, 82, 83].

The proceedings of the annual Symposium on Graph Drawing are published by Springer-Verlag in the LNCS series [88, 5]. Three special issues of journals dedicated to graph drawing have been recently assembled [14, 26, 25]. Additional special issues are planned for future Graph Drawing Symposia.

The author maintains a WWW page (`http://www.cs.brown.edu/people/rt/gd.html`) with pointers to graph drawing resources on the Web.

The rest of this paper is organized as follows: Section 3 overviews lower an upper bounds on fundamental drawing properties, such as area, and gives tradeoffs between them. Basic graph drawing terminology is reviewed in Section 2. Three-dimensional drawings are discussed in Section 4. Section 5 deals with methods for constraint satisfaction. Finally, experimental studies are reported in Section 6.

*Research supported in part by the National Science Foundation under grant CCR–9423847 and by the U.S. Army Research Office under grant DAAH04–96–1–0013.

2 Graph Drawing Glossary

First, we define some terminology on graphs pertinent to graph drawing:

n: number of vertices of the (di)graph being considered.

m: number of edges of the (di)graph being considered.

d: maximum vertex degree (i.e., number of incident edges) of the (di)graph being considered.

degree-k graph: graph with maximum degree $d \leq k$.

digraph: directed graph, i.e., graph with directed edges (drawn as arrows).

acyclic digraph: without directed cycles.

transitive edge: edge (u, v) of a digraph is transitive if there is a directed path from u to v not containing edge (u, v).

reduced digraph: without transitive edges.

source: vertex of a digraph without incoming edges.

sink: vertex of a digraph without outgoing edges.

st-digraph: acyclic digraph with exactly one source and one sink, joined by an edge (also called bipolar digraph).

connected graph: any two vertices are joined by a path.

biconnected graph: any two vertices are joined by two vertex-disjoint paths.

triconnected graph: any two vertices are joined by three vertex-disjoint paths.

tree: connected graph without cycles.

rooted tree: directed tree with a distinguished vertex, called the root, such that each vertex lies on a directed path to the root.

binary tree: rooted tree where each vertex has at most two incoming edges.

layered (di)graph: the vertices are partitioned into sets, called layers. A rooted tree can be viewed as a layered digraph where the layers are sets of vertices at the same distance from the root.

k-layered (di)graph: layered (di)graph with k layers.

In a drawing of a graph, vertices are represented by points (or by geometric figures such as circles or rectangles) and edges are represented by curves such that any two edges intersect at most in a finite number of points. Except for Section 4, which covers three-dimensional drawings, we consider drawings in the plane. The following types of drawings are defined:

polyline drawing: each edge is a polygonal chain (Fig. 1(a)).

straight-line drawing: each edge is a straight-line segment (Fig. 1(b)).

orthogonal drawing: each edge is a chain of horizontal and vertical segments (Fig. 1(c)).

bend: in a polyline drawing, point where two segments part of the same edge meet (Fig. 1(a)).

crossing: point where two edges intersect (Fig. 1(b)).

grid drawing: polyline drawing such that vertices, crossings and bends have integer coordinates.

planar drawing: no two edges cross (see Fig. 1(d)).

planar (di)graph: admits a planar drawing.

embedded (di)graph: planar (di)graph with a prespecified topological embedding (i.e., set of faces), which must be preserved in the drawing.

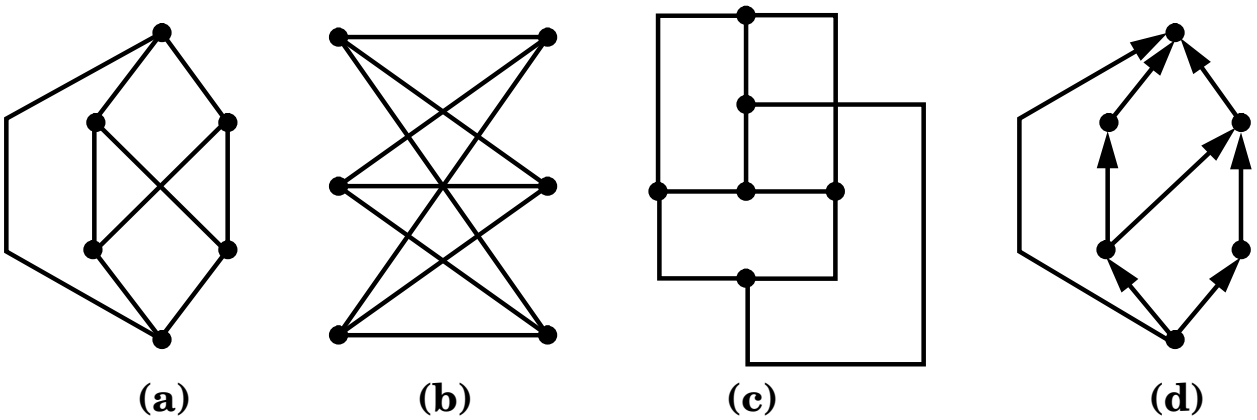


Figure 1: Types of drawings: (a) polyline drawing of $K_{3,3}$; (b) straight-line drawing of $K_{3,3}$; (c) orthogonal drawing of $K_{3,3}$; (d) planar upward drawing of an acyclic digraph.

upward drawing: drawing of a digraph where each edge is monotonically nondecreasing in the vertical direction (see Fig. 1(d)).

upward planar digraph: admits an upward planar drawing.

layered drawing: drawing of a layered graph such that vertices in the same layer are horizontally aligned (also called hierarchical drawing).

face: a region of the plane bounded by vertices and edges of a planar drawing.

convex drawing: planar straight-line drawing such that the boundary of each face is a convex polygon.

visibility drawing: drawing of a graph based on a geometric visibility relation. E.g., the vertices might be drawn as horizontal segments, and the edges associated with vertically visible segments.

proximity drawing: drawing of a graph based on a geometric proximity relation. E.g., a tree is drawn as the Euclidean minimum spanning tree of a set of points.

dominance drawing: upward drawing of an acyclic digraph such that there exists a directed path from vertex u to vertex v if and only if $x(u) \leq x(v)$ and $y(u) \leq y(v)$, where $x(\cdot)$ and $y(\cdot)$ denote the coordinates of a vertex.

hv-drawing: upward orthogonal straight-line drawing of a binary tree such that the drawings of the subtrees of each node are separated by a horizontal or vertical line.

Straight-line and orthogonal drawings are special cases of polyline drawings. Polyline drawings provide great flexibility since they can approximate drawings with curved edges. However, edges with more than two or three bends may be difficult to "follow" for the eye. Also, a system that supports editing of polyline drawings is more complicated than one limited to straight-line drawings. Hence, depending on the application, polyline or straight-line drawings may be preferred. If vertices are represented by points, orthogonal drawings exist only for graphs of maximum vertex degree 4.

3 Bounds and Tradeoffs on Drawing Properties

For various classes of graphs and drawing types, many universal/existential upper and lower bounds for specific drawing properties have been discovered. Such bounds typically exhibit trade-offs between drawing properties. A universal bound applies to all the graphs of a given class. An

existential bound applies to infinitely many graphs of the class.

Whenever we give bounds on the area or edge length, we assume that the drawing is constrained by some resolution rule that prevents it from being arbitrarily scaled down (e.g., requiring a grid drawing, or a minimum unit distance between any two vertices).

Bounds on the Area Table 1 summarizes selected universal upper bounds and existential lower bounds on the area of drawings of graphs.

Table 1: Universal upper bounds and existential lower bounds on the area of drawings of graphs. We denote with a an arbitrary constant such that $0 \leq a < 1$. We denote with b and c fixed constants such that $1 < b < c$.

Class of Graphs	Drawing Type	Area		Ref.
rooted tree	upward planar straight-line grid	$\Omega(n)$	$O(n \log n)$	[12, 79]
rooted tree	strictly upward planar straight-line grid	$\Omega(n \log n)$	$O(n \log n)$	[12]
degree-$O(n^a)$ rooted tree	upward planar polyline grid	$\Omega(n)$	$O(n)$	[38]
binary tree	upward planar orthogonal grid	$\Omega(n \log \log n)$	$O(n \log \log n)$	[38]
tree	planar straight-line grid	$\Omega(n)$	$O(n \log n)$	[12, 79]
degree-$O(n^a)$ tree	planar polyline grid	$\Omega(n)$	$O(n)$	[38]
degree-4 tree	planar orthogonal grid	$\Omega(n)$	$O(n)$	[93, 60]
planar graph	planar polyline grid	$\Omega(n^2)$	$O(n^2)$	[27, 28, 56]
planar graph	planar straight-line	$\Omega(c^{\rho n})$		[40]
planar graph	planar straight-line grid	$\Omega(n^2)$	$O(n^2)$	[19, 77]
triconnected planar graph	planar straight-line convex grid	$\Omega(n^2)$	$O(n^2)$	[56]
planar graph	planar orthogonal grid	$\Omega(n^2)$	$O(n^2)$	[3, 56, 81, 86]
planar degree-4 graph	orthogonal grid	$\Omega(n \log n)$	$O(n \log^2 n)$	[93, 60, 2]
upward planar digraph	upward planar grid straight-line	$O(b^n)$	$\Omega(c^n)$	[1, 28, 39]
reduced planar st-digraph	upward planar grid straight-line dominance	$\Omega(n^2)$	$O(n^2)$	[28]
upward planar digraph	upward planar grid polyline	$O(n^2)$	$\Omega(n^2)$	[27, 28]
general graph	polyline grid	$\Omega(n + \chi)$	$O((n + \chi)^2)$	

In general, the effect of bends on the area requirement is dual. On one hand, bends occupy space and hence negatively affect the area. On the other hand, bends may help in routing edges without using additional space.

The following comments apply to Table 1. Linear or almost-linear bounds on the area can be achieved for trees. See Table 4 for trade-offs between area and aspect-ratio in drawings of trees. Planar graphs admit planar drawings with quadratic area. However, the area requirement of planar straight-line drawings may be exponential if high angular resolution is also desired. Almost linear area can be instead achieved in nonplanar drawings of planar graphs, which have applications to VLSI circuits. Upward planar drawings provide an interesting trade-off between area and the total number of bends. Indeed, unless the digraph is reduced, the area can become exponential if a straight-line drawing is required. A quadratic area bound is achieved only at the expense of a linear number of bends.

Bounds on the Angular Resolution Tabe 2 summarizes selected universal lower bounds and existential upper bounds on the angular resolution of drawings of graphs.

Bounds on the Number of Bends Table 3 summarizes selected universal upper bounds and existential lower bounds on the total and maximum number of bends in orthogonal drawings. Some bounds are stated for $n \geq 5$ or $n \geq 7$ because the maximum number of bends is at least 2 for K_4 and at least 3 for the skeleton graph of an octahedron, in any planar orthogonal drawing

Table 2: Universal lower bounds and existential upper bounds on the angular resolution of drawings of graphs. We denote with c a fixed constant such that $c > 1$.

Class of Graphs	Drawing Type	Angular Resolution		Ref.
general graph	straight-line	$\Omega(\frac{1}{d^2})$	$O(\frac{\log d}{d^2})$	[35]
planar graph	straight-line	$\Omega(\frac{1}{d})$	$O(\frac{1}{d})$	[35]
planar graph	planar straight-line	$\Omega(\frac{1}{c^d})$	$O(\sqrt{\frac{\log d}{d^3}})$	[40, 65]

Table 3: Orthogonal drawings: universal upper bounds and existential lower bounds on the total and maximum number of bends. Notes: † $n \geq 7$; ‡ $n \geq 5$.

Class of Graphs	Drawing Type	Total No. Bends		Max No. Bends		Ref.
degree-4 graph†	orthogonal	$\geq n$	$\leq 2n+2$	≥ 2	≤ 2	[3]
planar degree-4 graph†	orthogonal planar	$\geq 2n-2$	$\leq 2n+2$	≥ 2	≤ 2	[3, 89]
embedded degree-4 graph	orthogonal planar	$\geq 2n-2$	$\leq \frac{12}{5}n+2$	≥ 3	≤ 3	[34, 64, 86, 89]
biconnected embedded degree-4 graph	orthogonal planar	$\geq 2n-2$	$\leq 2n+2$	≥ 3	≤ 3	[34, 64, 86, 89]
triconnected embedded degree-4 graph	orthogonal planar	$\geq \frac{4}{3}(n-1)+2$	$\leq \frac{3}{2}n+4$	≥ 2	≤ 2	[56]
embedded degree-3 graph‡	orthogonal planar	$\geq \frac{1}{2}n+1$	$\leq \frac{1}{2}n+1$	≥ 1	≤ 1	[56, 63]

Trade-Off Between Area and Aspect-Ratio The ability to construct area-efficient drawings is essential in practical visualization applications, where screen space is at a premium. However, achieving small area is not enough: e.g., it is easy to see that a drawing with high aspect-ratio may not be conveniently placed on a workstation screen, even if it has modest area. Hence, it is important to keep the aspect-ratio small. Ideally, one would like to obtain small area for any given aspect-ratio in a wide range. This would provide graphical user interfaces with the flexibility of fitting drawings in arbitrarily shaped windows.

A variety of trade-offs for the area and aspect-ratio arise even when drawing graphs with a simple structure, such as trees. Table 4 summarizes selected universal bounds that can be simultaneously achieved on the area and the aspect-ratio of various types of drawings of trees.

Table 4: Universal upper bounds that can be simultaneously achieved for the area and aspect-ratio in drawings of trees. We denote with a an arbitrary constant such that $0 \leq a < 1$.

Class of Graphs	Drawing Type	Area	Aspect-Ratio	Ref.
rooted tree	upward planar straight-line layered grid	$O(n^2)$	$O(1)$	[72]
rooted tree	upward planar straight-line grid	$O(n \log n)$	$O(n/\log n)$	[12, 79]
rooted degree-$O(1)$ tree	upward planar polyline grid	$O(n)$	$O(n^a)$	[38]
binary tree	upward planar orthogonal grid	$O(n \log\log n)$	$O(n \log\log n/\log^2 n)$	[38]
degree-4 tree	orthogonal grid	$O(n)$	$O(1)$	[93, 60]
degree-4 tree	orthogonal grid, leaves on convex hull	$O(n \log n)$	$O(1)$	[7]

While upward planar straight-line drawings are the most natural way of visualizing rooted trees, the existing drawing techniques are unsatisfactory with respect to either the area requirement or the aspect ratio. The situation is similar for orthogonal drawings. Regarding polyline drawings, linear area can be achieved with a prescribed aspect ratio [38]. However, experiments show that this is done at the expense of a somehow aesthetically unappealing drawing.

For non-upward drawings of trees, linear area and optimal aspect ratio are possible for planar orthogonal drawings, and a small (logarithmic) amount of extra area is needed if the leaves are constrained to be on the convex hull of the drawing (e.g., pins on the boundary of a VLSI circuit).

However, the non-upward drawing methods do not seem to yield aesthetically pleasing drawings, and are suited more for VLSI layout than for visualization applications.

Trade-Off Between Area and Angular Resolution Table 5 summarizes selected universal bounds that can be simultaneously achieved on the area and the angular resolution of drawings of graphs.

Table 5: Universal asymptotic upper bounds for the area and lower bounds for the angular resolution that can be simultaneously achieved in drawings of graphs. We denote with b and c fixed constants such that $b > 1$ and $c > 1$.

Class of Graphs	Drawing Type	Area	Angular Resolution	Ref.
planar graph	straight-line	$O(d^6 n)$	$\Omega(\frac{1}{d^2})$	[35]
planar graph	straight-line	$O(d^3 n)$	$\Omega(\frac{1}{d})$	[35]
planar graph	planar straight-line grid	$O(n^2)$	$\Omega(\frac{1}{n^2})$	[19, 77]
planar graph	planar straight-line	$O(b^n)$	$\Omega(\frac{1}{c^d})$	[65]
planar graph	planar polyline grid	$O(n^2)$	$\Omega(\frac{1}{d})$	[56]

Universal lower bounds on the angular resolution exist that depend only on the degree of the graph. Also, substantially better bounds can be achieved by drawing a planar graph with bends or in a nonplanar way.

Open Problems

- Determine the area requirement of (upward) planar straight-line drawings of trees. There is currently an $O(\log n)$ gap between the known upper and lower bounds (Table 1).
- Determine the area requirement of orthogonal (or, more generally, polyline) nonplanar drawings of planar graphs. There is currently an $O(\log n)$ gap between the known upper and lower bounds (Table 1).
- Close the gap between the $\Omega(\frac{1}{d^2})$ universal lower bound and the $O(\frac{\log d}{d^2})$ existential upper bound on the angular resolution of straight-line drawings of general graphs (Table 2).
- Close the gap between the $\Omega(\frac{1}{c^d})$ universal lower bound and the $O(\sqrt{\frac{\log d}{d^3}})$ existential upper bound on the angular resolution of planar straight-line drawings of planar graphs (Table 2).
- Determine the best possible aspect-ratio and area that can be simultaneously achieved for (upward) planar straight-line and orthogonal drawings of trees (Table 4).

4 Three Dimensional Drawings of Graphs

Recent advances in hardware and software technology for computer graphics open the possibility of displaying three-dimensional (3D) visualizations on a variety of low-cost workstations, and a handful of researchers (and film makers[1]) have begun to explore the possibilities of displaying graphs using this new technology Previous research on 3D graph drawing has focused on the development of visualization systems (see, e.g., [73, 76]). Much work needs to be done on the theoretical foundations of 3D graph drawing. Recent progress has been reported in [8, 9, 33, 43, 51, 62].

[1] An important plot element in the movie *Jurassic Park* involves a 3D virtual-reality traversal of a tree representing a Unix file system.

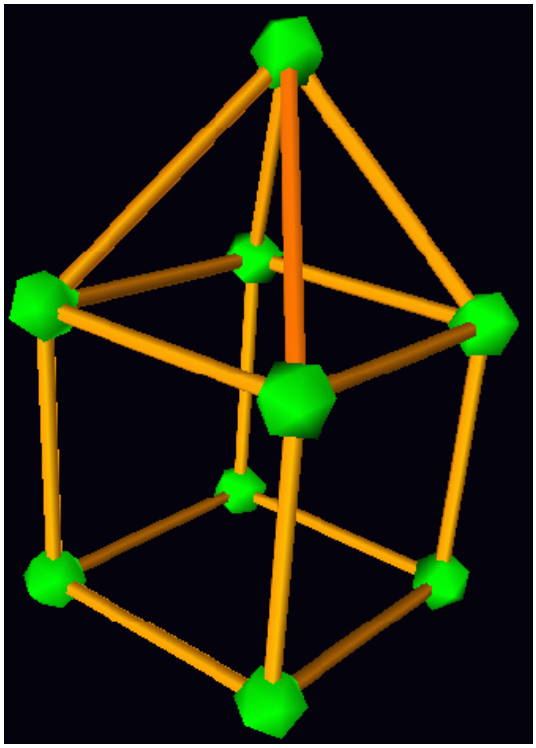


Figure 2: Example of a 3D convex drawing.

3D Convex Drawings A 3D convex drawing of a graph G is a realization of G by the skeleton of a 3D convex polytope (see Fig. 2. The well-known Steinitz's theorem says that a graph admits a 3D convex drawing if and only if it is planar and triconnected [80] (see also Grünbaum [42]), properties that can be verified in linear time (see, e.g., [48, 49]). Interestingly, it is a simple exercise to derive from the published proofs of Steinitz's theorem a cubic-time method for constructing 3D convex drawings in the real-RAM model [71]. Unfortunately, this approach seems to require at least exponential volume and an exponential number of bits to implement. Indeed, Onn and Sturmfels [68] show how to construct a 3D convex grid drawing within a cube of side $O(n^{169n^3})$.

Maxwell [67] (see also [10, 11, 94]) describes a mapping that transforms a 2D convex drawings with a certain "equilibrium stress property" into a 3D convex drawing. Further results on this transformation are given by Hopcroft and Kahn [50]. Eades and Garvan [31] show how to construct 3D convex drawings by combining the above transformation with the 2D-drawing method of Tutte [91, 92]. They also show that their drawings have exponential volume in the worst case. Smith (see [47]) claims a polynomial-time algorithm for constructing a 3D convex drawing inscribed in a sphere, with vertex coordinates represented by $O(n \log n)$-bit numbers, for an n-vertex graph known to be inscribable (which can be tested in linear time, e.g., for planar triangulations, due to a result of Dillencourt and Smith [30]). Das and Goodrich [17] present a linear-time algorithm for constructing a 3D convex drawing of a maximal planar graph such that the vertex coordinates are rational numbers that can be represented with a polynomial number of bits.

Chrobak, Goodrich and Tamassia [8] have recently shown how to construct in $O(n^{1.2})$ time a 3D convex drawing with $O(n)$ volume such that the vertex coordinates are represented by $O(n \log n)$-bit rational numbers and any two vertices are at distance at least one.

5 Constraint Satisfaction in Graph Drawing

Research in graph drawing has traditionally focused on algorithmic methods, where the drawing of the graph is generated according to a prespecified set of aesthetic criteria (such as planarity or area minimization) that are embodied in an algorithm. Although the algorithmic approach is computationally efficient, it does not naturally support constraints, i.e., requirements that the user may want to impose on the drawing of a specific graph (e.g., clustering or aligning a given set of vertices). Previous work has shown that only a rather limited constraint satisfaction capability can be added to existing drawing algorithms (see, e.g., [29, 84]).

Recently, several attempts have been made at developing languages for the specification of constraints and at devising techniques for graph drawing based on the resolution of systems of constraints (see, e.g., [20, 55, 66]). Eades and Lin [61] attempt at combining algorithmic and declarative methods in drawings of trees. Brandenburg presents a comprehensive approach to graph drawing based on graph grammars [4].

Visual Graph Drawing A visual approach to graph drawing, where the layout of a graph is pictorially specified "by example," is proposed by Cruz, Garg and Tamassia [15, 16]. Within this approach, a graph is stored in an object-oriented database, and its drawing is defined used recursive visual rules of the visual meta-language DOODLE [13]. The following types of drawings can be visually expressed in such a way that the system of constraints obtained from the application of the visual rules to the input graph can be solved in linear time:

- level drawings and box inclusion drawings of binary trees;
- Δ-drawings of series-parallel digraphs [1];
- polyline drawings [27], visibility drawings [85], and tessellation drawings [87] of upward planar digraphs (see Fig. 3).

In the rest of this section, we present visual programs for drawing a planar st-digraph, i.e., an embedded planar acyclic digraph with exactly one source and one sink, joined by an edge. Such digraphs play an important role in the theory of ordered sets since their transitive reductions are the covering digraphs of planar lattices [59]. Such visual programs can be easily modified to construct drawings of upward planar digraphs, which are known to be subgraphs of planar st-digraphs [58, 27].

We show in Figure 4 a complete visual program for tessellation representations. We assume that the vertices, edges, and faces of the input planar st-digraph G are database objects, where for each object o the following attributes describing the embedding are stored: left face $left(o)$, right face $right(o)$, bottom vertex $bot(o)$, and top vertex $top(o)$. Note that the value of each attribute is another database object.

Each rule defines the visual representation of a database object of a certain class (vertex, edge, and face). For tessellation representations, this is a horizontal segment for a vertex, a vertical segment for a face, and a rectangle for an edge. The visual notation in the rule for an object o includes:

- geometric figures that give the visual representation of object o, such as circles, segments, and rectangles;
- references to the visual representation of other objects given by attributes of o, denoted with dashed boxes labeled by the attribute;
- landmarks of the visual representations of o and of other referenced objects, shown as small squares with labels (e.g., MS, the "middle South" landmark, denotes the middle point of the bottom edge of a rectangle); and

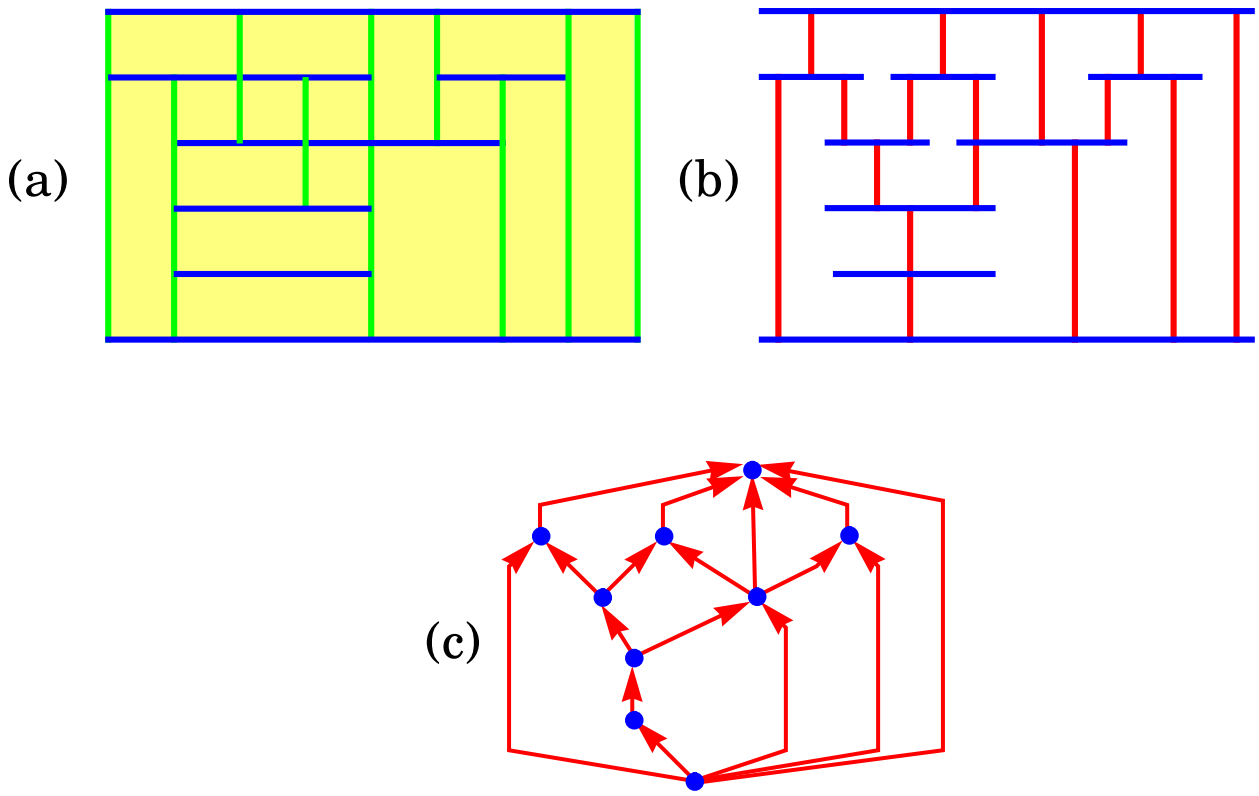


Figure 3: Drawings of a planar *st*-digraph: (a) tessellation drawing; (b) visibility drawing; (c) upward polyline drawing.

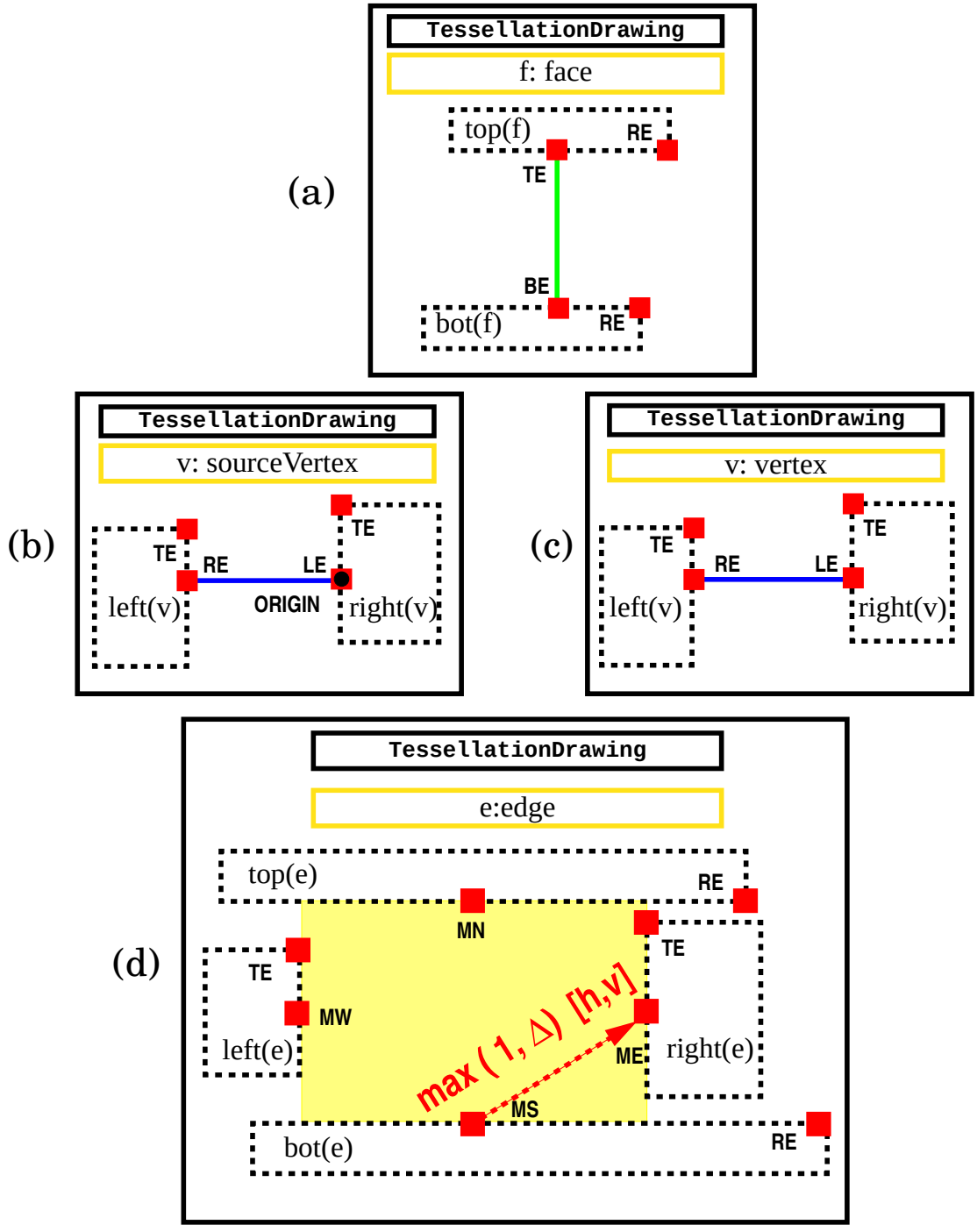


Figure 4: Visual rules for constucting a tessellation drawing of a planar *st*-digraph: (a) rule for a face; (b) special rule for the source vertex; (c) rule for a vertex; (d) rule for an edge.

- landmarks of the coordinate system, shown with small circles (e.g., ORIGIN denotes point $(0, 0)$);
- explicit constraints between landmarks, shown as arrows joining two landmarks with labels defining the constraint imposed on the coordinates of the landmarks (e.g., in rule (d), the dashed arrow with label max(1,Δ)[h,v] is an explicit constraint specifying minimum horizontal and vertical distance 1 from the "midpoint South" MS to the "midpoint East" of the rectangle);
- implicit constraints between landmarks, given by their horizontal or vertical alignment (e.g., in rule (d), the "midpoint East" ME of the rectangle representing edge e and the "top endpoint" TE of the referenced visual representation of the right face of e $right(e)$ must have the same x-coordinate because they are drawn vertically aligned).

Complete visual programs for visibility representations and upward polyline drawings are shown in Figures 5 and 6, respectively. In these two programs, the visual representation of the faces is a single point associated with landmark F. This point is invisible but contributes to the definition of the constraints. Also, the visual representation of an edge includes a visible portion (vertical segment for a visibility representation and polygonal chain with three segments for an upward polyline drawing) and an invisible portion drawn with a conventional "transparent color" (a rectangle or segment with shaded lines in the figures).

6 Experimental Graph Drawing

Many graph drawing algorithms have been implemented and used in practical applications. Most papers show sample outputs, and some also provide limited experimental results on small test suites (see, e.g., [18, 36, 37, 53, 55, 57] and the experimental papers in [88]). However, in order to evaluate the practical performance of a graph drawing algorithm in visualization applications, it is essential to perform extensive experimentations with input graphs derived from the application domain.

The performance of four planar straight-line drawing algorithms on 10,000 randomly generated maximal planar graphs is compared by Jones et al. [52].

Himsolt [45] presents a comparative study of twelve graph drawings algorithms based on various approaches. The experiments are conducted on 100 sample graphs with the graph drawing system *GraphEd* [46]. Many examples of drawings constructed by the algorithms are shown, and various objective and subjective evaluations on the aesthetic quality of the drawings produced are given.

Brandenburg and Rohrer [6] compare five "force-directed" methods for constructing straight-line drawings of general undirected graphs. The algorithms are tested on a a wide collection of examples and with different settings of the force parameters. The quality measures evaluated are crossings, edge length, vertex distribution, and running time. They also identify tradeoffs between the running time and the aesthetic quality of the drawings produced.

Jünger and Mutzel [54] investigate crossing minimization strategies for straight-line drawings of 2-layer graphs, and compare the performance of eight popular heuristics for this problem.

Experiments on Orthogonal Drawings In [22, 23] Di Battista et al. present an extensive experimental study comparing four general-purpose graph drawing algorithms. The four algorithms, denoted Bend-Stretch, Column, Giotto, and Pair, take as input general graphs (with no restrictions whatsoever on the connectivity, planarity, etc.) and construct orthogonal grid drawings, which are widely used in software and database visualization applications.

Algorithms Bend-Stretch and Giotto are based on a general approach where the drawing is incrementally specified in three phases: The first phase, *planarization*, determines the topology of

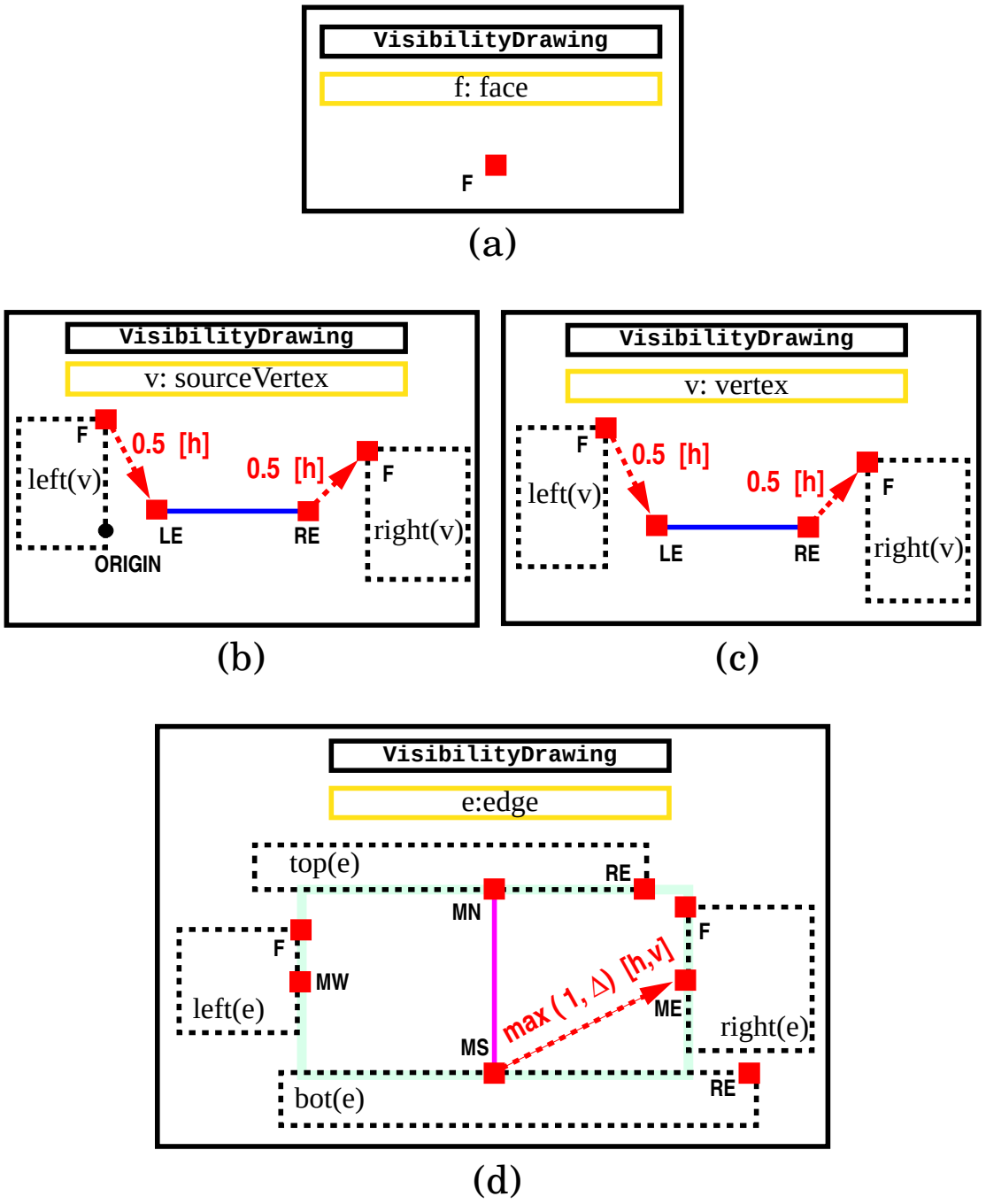


Figure 5: Visual rules for constucting a tessellation drawing of a planar *st*-digraph: (a) rule for a face; (b) special rule for the source vertex; (c) rule for a vertex; (d) rule for an edge.

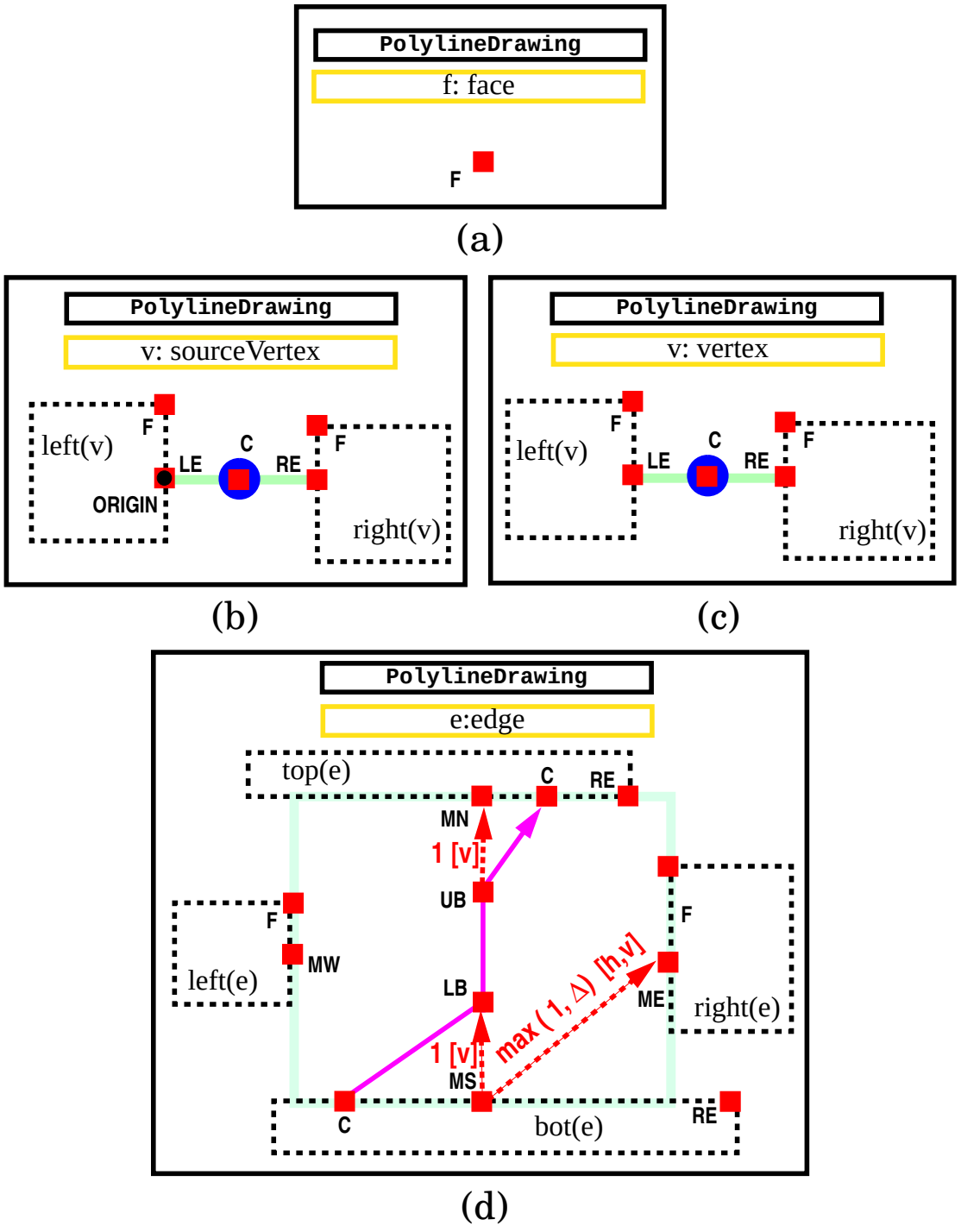


Figure 6: Visual rules for constucting a tessellation drawing of a planar *st*-digraph: (a) rule for a face; (b) special rule for the source vertex; (c) rule for a vertex; (d) rule for an edge.

the drawing. The second phase, *orthogonalization*, computes an orthogonal shape for the drawing. The third phase, *compaction*, produces the final drawing. This approach allows homogeneous treatment of a wide range of diagrammatic representations, aesthetics and constraints (see, e.g., [57, 84, 90]) and has been successfully used in industrial tools. The main difference between the two algorithms is in the orthogonalization phase: Algorithm Giotto uses a network-flow method that guarantees the minimum number of bends but has quadratic time-complexity [81]. Algorithm Bend-Stretch adopts the "bend-stretching" heuristic [86] that only guarantees a constant number of bends on each edge but runs in linear time.

Algorithm Column is an extension of the orthogonal drawing algorithm by Biedl and Kant [3] to graphs of arbitrary vertex degree. The orthogonal grid drawing is incrementally constructed by adding the vertices one at a time. Namely, at each step a vertex v is added plus the edges connecting v to previously added vertices. Some columns of the grid are "reserved" to draw the remaining incident edges of v. Concerning the position of v, since one row is used for each vertex, the y-coordinate is immediately given by the order of visit of v, and the x-coordinate is the one of the reserved column of the incident edge of v that minimizes the number of bends introduced by the new edges. Algorithm Pair is an extension of the orthogonal drawing algorithm by Papakostas and Tollis [69, 70] to graphs of arbitrary vertex degree.

Examples of "typical" drawings generated by Bend-Stretch, Column, Giotto, and Pair are shown in Figures 7.

The test data (available on the Internet) are 11,582 graphs, ranging from 10 to 100 vertices, generated from a core set of 112 graphs used in "real-life" software engineering and database applications. The experiments provide a detailed quantitative evaluation of the performance of the four algorithms and show that they exhibit trade-offs between "aesthetic" properties (e.g., crossings, bends, edge length) and running time. For example, Fig. 8 shows the average area number of crossings, and CPU time. The observed practical behavior of the algorithms is consistent with their theoretical properties. Namely, Giotto outperforms the other algorithms for most quality measures but is considerably slower than Column and Pair.

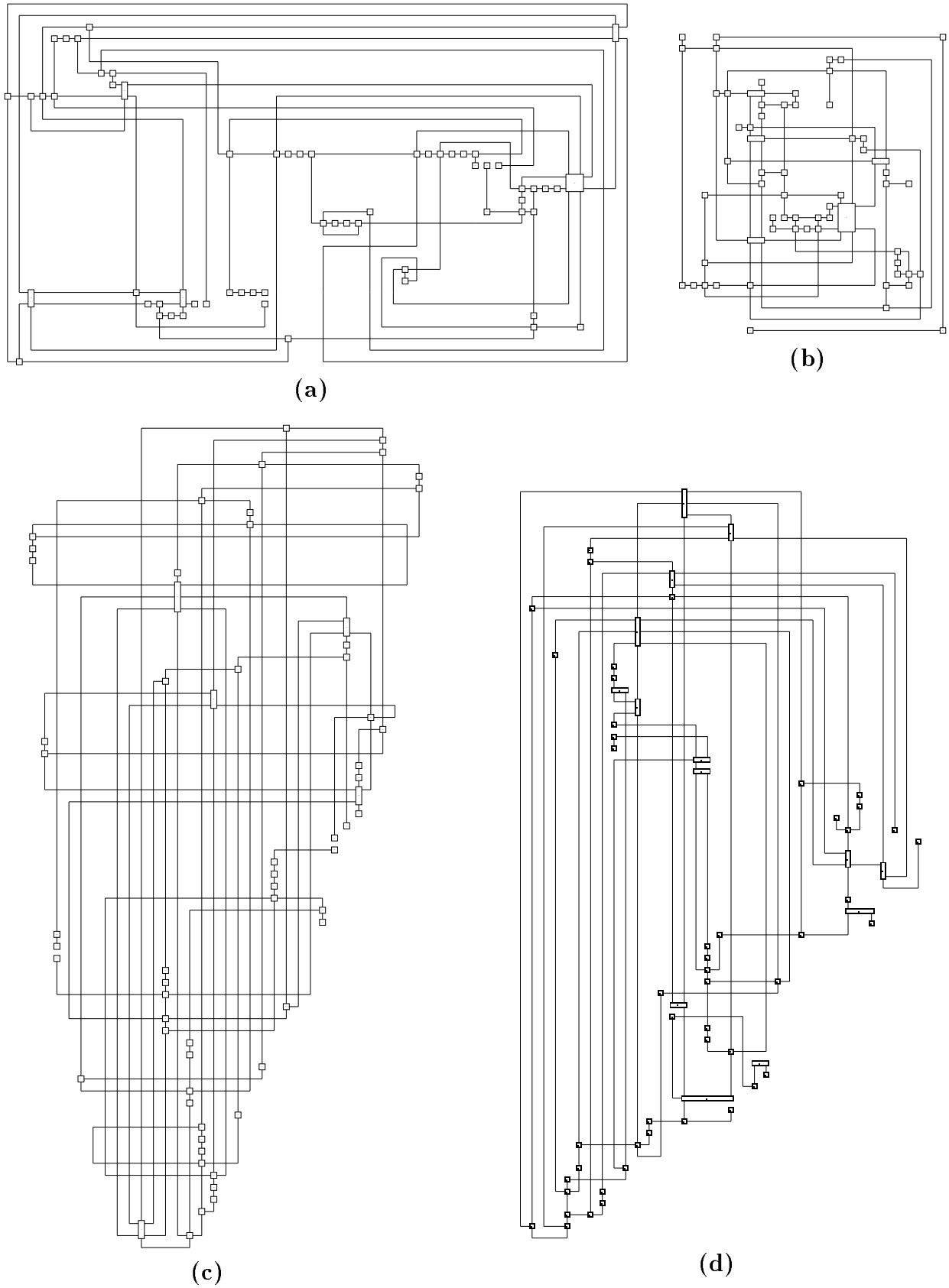
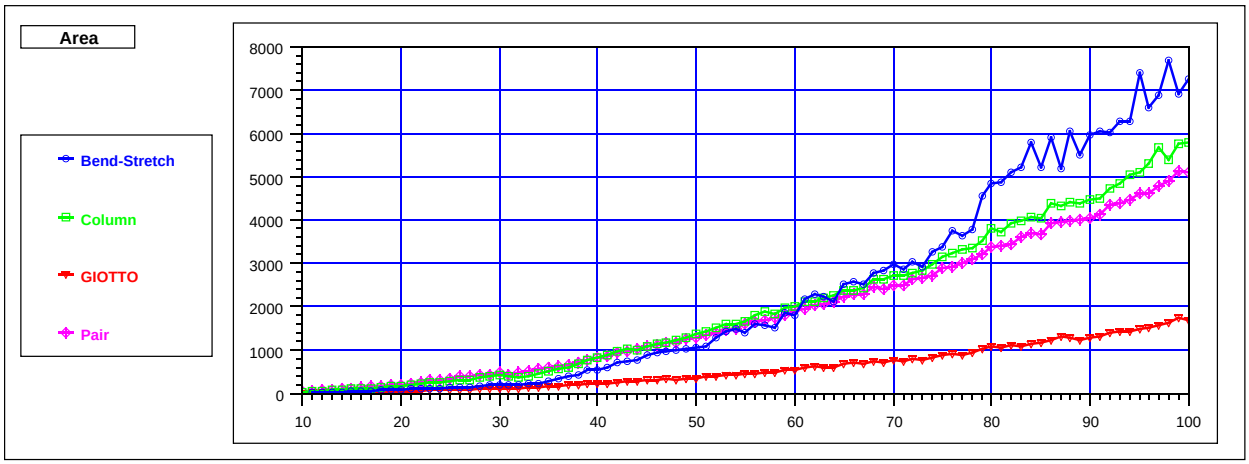
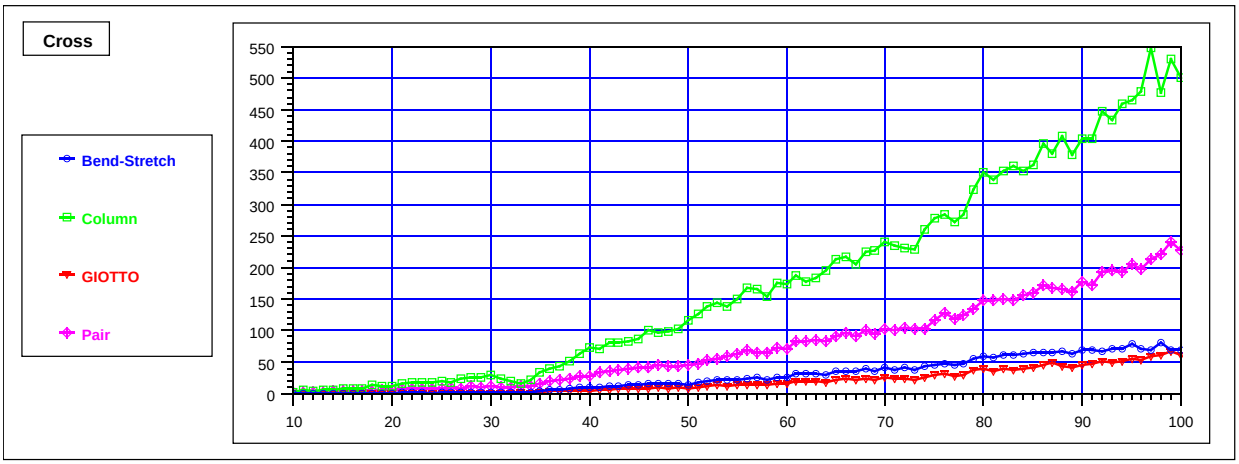


Figure 7: Drawings of the same 63-vertex graph produced by algorithms **(a)** Bend-Stretch, **(b)** Giotto, **(c)** Column, and **(d)** Pair, respectively.

(a)



(b)



(c)

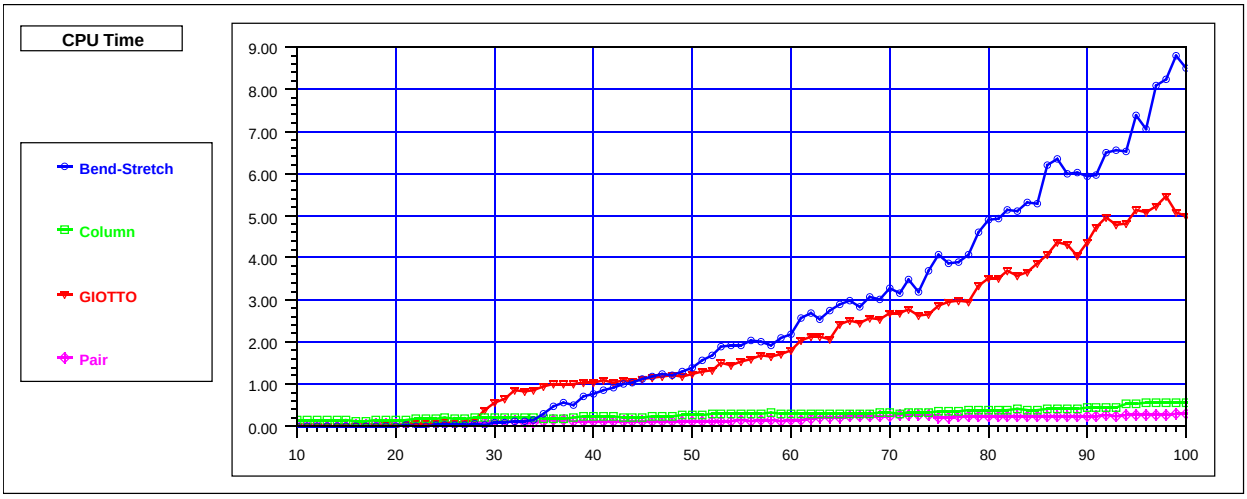


Figure 8: **(a)** Average area versus number of vertices. **(b)** Average number of crossings versus number of vertices. **(c)** Average CPU time (seconds) versus number of vertices.

References

[1] P. Bertolazzi, R. F. Cohen, G. Di Battista, R. Tamassia, and I. G. Tollis. How to draw a series-parallel digraph. *Internat. J. Comput. Geom. Appl.*, 4:385–402, 1994.

[2] S. N. Bhatt and F. T. Leighton. A framework for solving VLSI graph layout problems. *J. Comput. Syst. Sci.*, 28:300–343, 1984.

[3] T. Biedl and G. Kant. A better heuristic for orthogonal graph drawings. In *Proc. 2nd Annu. European Sympos. Algorithms (ESA '94)*, volume 855 of *Lecture Notes in Computer Science*, pages 24–35. Springer-Verlag, 1994.

[4] F. J. Brandenburg. Designing graph drawings by layout graph grammars. In R. Tamassia and I. G. Tollis, editors, *Graph Drawing (Proc. GD '94)*, volume 894 of *Lecture Notes in Computer Science*, pages 416–427. Springer-Verlag, 1995.

[5] F. J. Brandenburg, editor. *Graph Drawing (Proc. GD '95)*, volume 1027 of *Lecture Notes in Computer Science*. Springer-Verlag, 1996.

[6] Franz J. Brandenburg and Christoph Rohrer. An experimental comparison of force-directed and randomized graph drawing algorithms. In F. J. Brandenburg, editor, *Graph Drawing (Proc. GD '95)*, volume 1027 of *Lecture Notes in Computer Science*. Springer-Verlag, 1996.

[7] R. P. Brent and H. T. Kung. On the area of binary tree layouts. *Inform. Process. Lett.*, 11:521–534, 1980.

[8] M. Chrobak, M. T. Goodrich, and R. Tamassia. Convex drawings of graphs in two and three dimensions. In *Proc. 12th Annu. ACM Sympos. Comput. Geom.*, 1996. To appear.

[9] R. F. Cohen, P. Eades, T. Lin, and F. Ruskey. Three-dimensional graph drawing. In R. Tamassia and I. G. Tollis, editors, *Graph Drawing (Proc. GD '94)*, volume 894 of *Lecture Notes in Computer Science*, pages 1–11. Springer-Verlag, 1995.

[10] R. Connelly. Rigidity and energy. *Invent. Math.*, 66:11–33, 1982.

[11] H. Crapo and W. Whitely. Statics of frameworks and motions of panel structures, a projective geometric introduction. *Structural Topology*, 6:42–82, 1982.

[12] P. Crescenzi, G. Di Battista, and A. Piperno. A note on optimal area algorithms for upward drawings of binary trees. *Comput. Geom. Theory Appl.*, 2:187–200, 1992.

[13] I. F. Cruz. DOODLE: A visual language for object-oriented databases. In *Proc. ACM SIGMOD*, pages 71–80, 1992.

[14] I. F. Cruz and P. Eades, editors. *Special Issue on Graph Visualization*, volume 6:3 of *J. Visual Languages and Computing*. 1995.

[15] I. F. Cruz and A. Garg. Drawing graphs by example efficiently: Trees and planar acyclic digraphs. In R. Tamassia and I. G. Tollis, editors, *Graph Drawing (Proc. GD '94)*, volume 894 of *Lecture Notes in Computer Science*, pages 404–415. Springer-Verlag, 1995.

[16] I. F. Cruz, A. Garg, and R. Tamassia. Efficient constraint resolution in visual graph drawing. Manuscript, Dept. of Computer Sci., Brown University, 1996.

[17] Gautam Das and Michael T. Goodrich. On the complexity of approximating and illuminating three-dimensional convex polyhedra. In *Proc. 4th Workshop Algorithms Data Struct.*, volume 955 of *Lecture Notes in Computer Science*, pages 74–85. Springer-Verlag, 1995.

[18] R. Davidson and D. Harel. Drawing graphs nicely using simulated annealing. Technical report, Department of Applied Mathematics and Computer Science, The Weizmann Institute of Science, Rehovot, 1989.

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

[20] E. Dengler, M. Friedell, and J. Marks. Constraint-driven diagram layout. In *Proc. IEEE Sympos. on Visual Languages (VL '93)*, pages 330–335, 1993.

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

[22] G. Di Battista, A. Garg, G. Liotta, R. Tamassia, E. Tassinari, and F. Vargiu. An experimental comparison of three graph drawing algorithms. In *Proc. 11th Annu. ACM Sympos. Comput. Geom.*, pages 306–315, 1995.

[23] G. Di Battista, A. Garg, G. Liotta, R. Tamassia, E. Tassinari, and F. Vargiu. An experimental comparison of four graph drawing algorithms. Submitted to *Computational Geometry: Theory and Applications*, 1995.

[24] G. Di Battista, W. Lenhart, and G. Liotta. Proximity drawability: a survey. In R. Tamassia and I. G. Tollis, editors, *Graph Drawing (Proc. GD '94)*, volume 894 of *Lecture Notes in Computer Science*, pages 328–339. Springer-Verlag, 1995.

[25] G. Di Battista and R. Tamassia, editors. *Special Issue on Geometric Representations of Graphs.* Comput. Geom. Theory Appl. To appear.

[26] G. Di Battista and R. Tamassia, editors. *Special Issue on Graph Drawing.* Algorithmica. To appear.

[27] G. Di Battista and R. Tamassia. Algorithms for plane representations of acyclic digraphs. *Theoret. Comput. Sci.*, 61:175–198, 1988.

[28] G. Di Battista, R. Tamassia, and I. G. Tollis. Area requirement and symmetry display of planar upward drawings. *Discrete Comput. Geom.*, 7:381–401, 1992.

[29] G. Di Battista, R. Tamassia, and I. G. Tollis. Constrained visibility representations of graphs. *Inform. Process. Lett.*, 41:1–7, 1992.

[30] M. B. Dillencourt and W. D. Smith. A linear-time algorithm for testing the inscribability of trivalent polyhedra. In *Proc. 8th Annu. ACM Sympos. Comput. Geom.*, pages 177–185, 1992.

[31] P. Eades and P. Garvan. Drawing stressed planar graphs in three dimensions. In F. J. Brandenburg, editor, *Graph Drawing (Proc. GD '95)*, volume 1027 of *Lecture Notes in Computer Science*. Springer-Verlag, 1996.

[32] P. Eades and X. Lin. How to draw a directed graph. In *Proc. IEEE Workshop on Visual Languages (VL'89)*, pages 13–17, 1989.

[33] P. Eades, C. Stirk, and S. Whitesides. The techniques of Komolgorov and Bardzin for three dimensional orthogonal graph drawings. Manuscript, Dept. of Computer Sci., Univ. of Newcastle, 1995.

[34] S. Even and G. Granot. Rectilinear planar drawings with few bends in each edge. Technical Report 797, Computer Science Dept., Technion, 1994.

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

[36] T. Fruchterman and E. Reingold. Graph drawing by force-directed placement. *Softw. – Pract. Exp.*, 21(11):1129–1164, 1991.

[37] E. R. Gansner, S. C. North, and K. P. Vo. DAG – A program that draws directed graphs. *Softw. – Pract. Exp.*, 18(11):1047–1062, 1988.

[38] A. Garg, M. T. Goodrich, and R. Tamassia. Area-efficient upward tree drawings. In *Proc. 9th Annu. ACM Sympos. Comput. Geom.*, pages 359–368, 1993.

[39] A. Garg and R. Tamassia. Efficient computation of planar straight-line upward drawings. In *Graph Drawing '93 (Proc. ALCOM Workshop on Graph Drawing)*, Paris, France, 1993.

[40] A. Garg and R. Tamassia. Planar drawings and angular resolution: Algorithms and bounds. In *Proc. 2nd Annu. European Sympos. Algorithms (ESA '94)*, volume 855 of *Lecture Notes in Computer Science*, pages 12–23. Springer-Verlag, 1994.

[41] A. Garg and R. Tamassia. Upward planarity testing. *Order*, 12:109–133, 1995.

[42] B. Grünbaum. *Convex Polytopes.* Wiley, New York, NY, 1967.

[43] S. M. Hashemi and I. Rival. Upward drawings to fit surfaces. In *Order, Algorithms, and Applications (Proc. ORDAL '94)*, volume 831 of *Lecture Notes in Computer Science*, pages 53–58. Springer-Verlag, 1994.

[44] X. He and M.-Y. Kao. Regular edge labelings and drawings of planar graphs. In R. Tamassia and I. G. Tollis, editors, *Graph Drawing (Proc. GD '94)*, volume 894 of *Lecture Notes in Computer Science*, pages 96–103. Springer-Verlag, 1995.

[45] M. Himsolt. Comparing and evaluating layout algorithms within GraphEd. *J. Visual Languages and Computing*, 6(3), 1995. (special issue on Graph Visualization, edited by I. F. Cruz and P. Eades).

[46] M. Himsolt. GraphEd: a graphical platform for the implementation of graph algorithms. In R. Tamassia and I. G. Tollis, editors, *Graph Drawing (Proc. GD '94)*, volume 894 of *Lecture Notes in Computer Science*, pages 182–193. Springer-Verlag, 1995.

[47] C. D. Hodgson, I. Rivin, and W. D. Smith. A characterization of convex hyperbolic polyhedra and of convex polyhedra inscribed in the sphere. *Bull. (New Series) of the AMS*, 27(2):246–251, 1992.

[48] J. Hopcroft and R. E. Tarjan. Dividing a graph into triconnected components. *SIAM J. Comput.*, 2:135–158, 1973.

[49] J. Hopcroft and R. E. Tarjan. Efficient planarity testing. *J. ACM*, 21(4):549–568, 1974.

[50] J. E. Hopcroft and P. J. Kahn. A paradigm for robust geometric algorithms. *Algorithmica*, 7:339–380, 1992.

[51] T. Jéron and C. Jard. 3D layout of reachability graphs of communicating processes. In R. Tamassia and I. G. Tollis, editors, *Graph Drawing (Proc. GD '94)*, volume 894 of *Lecture Notes in Computer Science*, pages 25–32. Springer-Verlag, 1995.

[52] S. Jones, P. Eades, A. Moran, N. Ward, G. Delott, and R. Tamassia. A note on planar graph drawing algorithms. Technical Report 216, Department of Computer Science, University of Queensland, 1991.

[53] M. Juenger and P. Mutzel. Maximum planar subgraphs and nice embeddings: Practical layout tools. *Algorithmica.* (special issue on Graph Drawing, edited by G. Di Battista and R. Tamassia, to appear).

[54] Michael Jünger and Petra Mutzel. Exact and heuristic algorithms for 2-layer straightline crossing minimization. In F. J. Brandenburg, editor, *Graph Drawing (Proc. GD '95)*, volume 1027 of *Lecture Notes in Computer Science.* Springer-Verlag, 1996.

[55] T. Kamada. *Visualizing Abstract Objects and Relations.* World Scientific Series in Computer Science, 1989.

[56] G. Kant. Drawing planar graphs using the canonical ordering. *Algorithmica.* (special issue on

Graph Drawing, edited by G. Di Battista and R. Tamassia, to appear).

[57] G. Kant. *Algorithms for Drawing Planar Graphs.* PhD thesis, Dept. Comput. Sci., Univ. Utrecht, Utrecht, Netherlands, 1993.

[58] D. Kelly. Fundamentals of planar ordered sets. *Discrete Math.*, 63:197–216, 1987.

[59] D. Kelly and I. Rival. Planar lattices. *Canad. J. Math.*, 27(3):636–665, 1975.

[60] C. E. Leiserson. Area-efficient graph layouts (for VLSI). In *Proc. 21st Annu. IEEE Sympos. Found. Comput. Sci.*, pages 270–281, 1980.

[61] T. Lin and P. Eades. Integration of declarative and algorithmic approaches for layout creation. In R. Tamassia and I. G. Tollis, editors, *Graph Drawing (Proc. GD '94)*, volume 894 of *Lecture Notes in Computer Science*, pages 376–387. Springer-Verlag, 1995.

[62] Giuseppe Liotta and Giuseppe Di Battista. Computing proximity drawings of trees in the 3-dimemsional space. In *Proc. 4th Workshop Algorithms Data Struct.*, volume 955 of *Lecture Notes in Computer Science*, pages 239–250. Springer-Verlag, 1995.

[63] Y. Liu, P. Marchioro, R. Petreschi, and B. Simeone. Theoretical results on at most 1-bend embeddability of graphs. Technical report, Dipartimento di Statistica, Univ. di Roma "La Sapienza", 1990.

[64] Y. Liu, A. Morgana, and B. Simeone. General theoretical results on rectilinear embeddability of graphs. *Acta Math. Appl. Sinica*, 7:187–192, 1991.

[65] S. Malitz and A. Papakostas. On the angular resolution of planar graphs. *SIAM J. Discrete Math.*, 7:172–183, 1994.

[66] J. Marks. A formal specification for network diagrams that facilitates automated design. *Journal of Visual Languages and Computing*, 2:395–414, 1991.

[67] J. C. Maxwell. On reciprocal figures and diagrams of forces. *Phil. Mag. Ser.*, 27:250–261, 1864.

[68] S. Onn and B. Sturmfels. A quantitative Steinitz' theorem. *Beiträge zur Algebra und Geometrie / Contributions to Algebra and Geometry*, 35:125–129, 1994.

[69] A. Papakostas and I. G. Tollis. Improved algorithms and bounds for orthogonal drawings. In R. Tamassia and I. G. Tollis, editors, *Graph Drawing (Proc. GD '94)*, volume 894 of *Lecture Notes in Computer Science*, pages 40–51. Springer-Verlag, 1995.

[70] A. Papakostas and I. G. Tollis. Improved algorithms and bounds for orthogonal drawings. Technical report, 1995.

[71] F. P. Preparata and M. I. Shamos. *Computational Geometry: An Introduction.* Springer-Verlag, New York, NY, 1985.

[72] E. Reingold and J. Tilford. Tidier drawing of trees. *IEEE Trans. Softw. Eng.*, SE-7(2):223–228, 1981.

[73] S. P. Reiss. An engine for the 3D visualization of program information. *J. Visual Languages and Computing*, 6(3), 1995. (special issue on Graph Visualization, edited by I. F. Cruz and P. Eades).

[74] I. Rival. Graphical data structures for ordered sets. In I. Rival, editor, *Algorithms and Order*, pages 3–31. Kluwer Academic Publishers, 1989.

[75] I. Rival. Reading, drawing, and order. In I. G. Rosenberg and G. Sabidussi, editors, *Algebras and Orders*, pages 359–404. Kluwer Academic Publishers, 1993.

[76] G. G. Robertson, J. D. Mackinlay, and S. K. Card. Cone trees: Animated 3d visualizations of hierarchical information. In *Proc. CHI*, pages 189–193, 1991.

[77] W. Schnyder. Embedding planar graphs on the grid. In *Proc. 1st ACM-SIAM Sympos. Discrete Algorithms*, pages 138–148, 1990.

[78] F. Shahrokhi, L. A. Székely, and I. Vrt'o. Crossing numbers of graphs, lower bound techniques and algorithms: a survey. In R. Tamassia and I. G. Tollis, editors, *Graph Drawing (Proc. GD '94)*, volume 894 of *Lecture Notes in Computer Science*, pages 131–142. Springer-Verlag, 1995.

[79] Y. Shiloach. *Arrangements of Planar Graphs on the Planar Lattice.* PhD thesis, Weizmann Institute of Science, 1976.

[80] E. Steinitz and H. Rademacher. *Vorlesungen über die Theorie der Polyeder.* Julius Springer, Berlin, Germany, 1934.

[81] R. Tamassia. On embedding a graph in the grid with the minimum number of bends. *SIAM J. Comput.*, 16(3):421–444, 1987.

[82] R. Tamassia. Drawing algorithms for planar st-graphs. *Australasian Journal of Combinatorics*, 2:217–235, 1990.

[83] R. Tamassia. Planar orthogonal drawings of graphs. In *Proc. IEEE Internat. Sympos. on Circuits and Systems*, 1990.

[84] R. Tamassia, G. Di Battista, and C. Batini. Automatic graph drawing and readability of diagrams. *IEEE Trans. Syst. Man Cybern.*, SMC-18(1):61–79, 1988.

[85] R. Tamassia and I. G. Tollis. A unified approach to visibility representations of planar graphs. *Discrete Comput. Geom.*, 1(4):321–341, 1986.

[86] R. Tamassia and I. G. Tollis. Planar grid embedding in linear time. *IEEE Trans. on Circuits and Systems*, CAS-36(9):1230–1234, 1989.

[87] R. Tamassia and I. G. Tollis. Tessellation representations of planar graphs. In *Proc. 27th Allerton Conf. Commun. Control Comput.*, pages 48–57, 1989.

[88] R. Tamassia and I. G. Tollis, editors. *Graph Drawing (Proc. GD '94)*, volume 894 of *Lecture Notes in Computer Science.* Springer-Verlag, 1995.

[89] R. Tamassia, I. G. Tollis, and J. S. Vitter. Lower bounds for planar orthogonal drawings of graphs. *Inform. Process. Lett.*, 39:35–40, 1991.

[90] H. Trickey. Drag: A graph drawing system. In *Proc. Internat. Conf. on Electronic Publishing*, pages 171–182. Cambridge University Press, 1988.

[91] W. T. Tutte. Convex representations of graphs. *Proceedings London Mathematical Society*, 10(3):304–320, 1960.

[92] W. T. Tutte. How to draw a graph. *Proceedings London Mathematical Society*, 13(3):743–768, 1963.

[93] L. Valiant. Universality considerations in VLSI circuits. *IEEE Trans. Comput.*, C-30(2):135–140, 1981.

[94] W. Whitney. Motions and stresses of projected polyhedra. *Structural Topology*, 7:13–38, 1982.