

- [34] Schnyder, W., Embedding planar graphs on the grid, in: *Proc. 1st Annual ACM-SIAM Symp. on Discr. Alg.*, San Francisco, 1990, pp. 138–147.
- [35] Storer, J.A., On minimal node-cost planar embeddings, *Networks* 14 (1984), pp. 181–212.
- [36] Tamassia, R., On embedding a graph in the grid with the minimum number of bends, *SIAM J. Comput.* 16 (1987), pp. 421–444.
- [37] Tamassia, R., and I.G. Tollis, Planar Grid Embedding in Linear Time, *IEEE Trans. on Circuits and Systems*, CAS-36 (1989), pp. 1230–1234.
- [38] Tamassia, R., and I.G. Tollis, A unified approach to visibility representations of planar graphs, *Discr. and Comp. Geometry* 1 (1986), pp. 321–341.
- [39] Tamassia, R., I.G. Tollis and J.S. Vitter, Lower bounds for planar orthogonal drawings of graphs, *Inf. Proc. Letters* 39 (1991), pp. 35–40.
- [40] Thomassen, C., Planarity and duality of finite and infinite planar graphs, *J. Comb. Theory Series B*, Vol. 29 (1980), pp. 244–271.
- [41] Tutte, W.T., Convex representations of graphs, *Proc. London Math. Soc.* 10 (1960), pp. 302–320.
- [42] Tutte, W.T., How to draw a graph, *Proc. London Math. Soc.* 13 (1963), pp. 743–768.
- [43] Wagner, K., Bemerkungen zum vierfarbenproblem, *Jber. Deutsch. Math.-Verein* 46 (1936), pp. 26–32.

- [18] A. Garg, and R. Tamassia, *On the Computational Complexity of Upward and Rectilinear Planarity Testing*, Tech. Report CS-94-10, Dept. of Comp. Science, Brown University, Providence, 1994.
- [19] Haandel, F. van, *Straight Line Embeddings on the Grid*, Dept. of Comp. Science, M.Sc. Thesis, no. INF/SCR-91-19, Utrecht University, 1991.
- [20] He, X., On finding the rectangular duals of planar triangulated graphs, *SIAM J. Comput.*, 1993, to appear.
- [21] Hopcroft, J., and R.E. Tarjan, Dividing a graph into triconnected components, *SIAM J. Comput.* 2 (1973), pp. 135–158.
- [22] Hopcroft, J., and R.E. Tarjan, Efficient planarity testing, *J. ACM* 21 (1974), pp. 549–568.
- [23] Jayakumar, R., K. Thulasiraman, and M.N.S. Swamy, Planar embedding: linear-time algorithms for vertex placement and edge ordering, *IEEE Trans. on Circuits and Systems* 35 (1988), pp. 334–344.
- [24] Kant, G., Hexagonal Grid Drawings, in: E.W. Mayr (Ed.), *Proc. 18th Intern. Workshop on Graph-Theoretic Concepts in Comp. Science WG'92*, Lecture Notes in Comp. Science, Springer-Verlag, 1992, pp. 263–276.
- [25] Kant, G., *Algorithms for Drawing Planar Graphs*, PhD thesis, Dept. of Computer Science, Utrecht University, 1993.
- [26] Kant, G., The Planar Triconnectivity Augmentation Problem, submitted to *Theoretical Computer Science*, 1993.
- [27] Kant, G., A More Compact Visibility Representation, in: J. van Leeuwen (Ed.), *Proc. 19th Intern. Workshop on Graph-Theoretic Concepts in Comp. Science WG'93*, Lecture Notes in Comp. Science, Springer-Verlag, 1993 (to appear).
- [28] Kant, G., and X. He, Two Algorithms for Finding Rectangular Duals of Planar Graphs, in: J. van Leeuwen (Ed.), *Proc. 19th Intern. Workshop on Graph-Theoretic Concepts in Comp. Science WG'93*, Lecture Notes in Comp. Science, Springer-Verlag, 1993 (to appear).
- [29] Lempel, A., S. Even and I. Cederbaum, An algorithm for planarity testing of graphs, *Theory of Graphs, Int. Symp. Rome* (1966), pp. 215–232.
- [30] Lin, Y.-L., and S.S. Skiena, *Complexity Aspects of Visibility Graphs*, Manuscript, Dept. of Comp. Science, State Univ. of New York, Stony Brook, 1992.
- [31] Malitz, S., and A. Papakostas, On the Angular Resolution of Planar Graphs, in: *Proc. 24th Annual ACM Symp. Theory of Computing*, Victoria, 1992.
- [32] Nummenmaa, J., Constructing compact rectilinear planar layouts using canonical representation of planar graphs, *Theoret. Comp. Science* 99 (1992), pp. 213–230.
- [33] Rosenstiehl, P., and R.E. Tarjan, Rectilinear planar layouts and bipolar orientations of planar graphs, *Discr. and Comp. Geometry* 1 (1986), pp. 343–353.

- [4] Chiba, N., T. Nishizeki, S. Abe and T. Ozawa, A linear algorithm for embedding planar graphs using PQ-trees, *J. of Computer and System Sciences*, Vol. 30 (1985), pp. 54–76.
- [5] Chiba, N., T. Yamanouchi, and T. Nishizeki, Linear algorithms for convex drawings of planar graphs, in: *Progress in Graph Theory*, J.A. Bondy and U.S.R. Murty (Eds.), Academic Press, 1984, pp. 153–173.
- [6] Cohen, R.F., G. Di Battista, R. Tamassia, I.G. Tollis and P. Bertolazzi, A Framework for dynamic graph drawing, in: *Proc. 8th Annual ACM Symp. on Computational Geometry*, Berlin, 1992, pp. 261–270.
- [7] Chrobak, M., and T.H. Payne, *A Linear Time Algorithm for Drawing Planar Graphs on the Grid*, Tech. Rep. UCR-CS-90-2, Dept. of Math. and Comp. Science, University of California at Riverside, 1990.
- [8] Chrobak, M., and G. Kant, *Convex Grid Drawings of 3-Connected Planar Graphs*, Tech. Rep. RUU-CS-93-45, Dept. of Computer Science, Utrecht University, 1993.
- [9] G. Di Battista, P. Eades, R. Tamassia, and I.G. Tollis Algorithms for Automatic Graph Drawing: An Annotated Bibliography, to appear in *Comp. Geometry: Theory and Applications*, Preliminary version also available via anonymous ftp from `wilma.cs.brown.edu` (128.148.33.66), files `/pub/gdbiblio.tex.Z` and `/pub/gdbiblio.ps.Z`.
- [10] Di Battista, G., and R. Tamassia, Incremental planarity testing, in: *Proc. 30th Annual IEEE Symp. on Found. of Comp. Science*, North Carolina, 1989, pp. 436–441.
- [11] Di Battista, G., R. Tamassia and I.G. Tollis, Area requirement and symmetry display in drawing graphs, *Discr. and Comp. Geometry* 7 (1992), pp. 381–401.
- [12] Di Battista, G., and L. Vismara, Angles of Planar Triangular Graphs, extended abstract in: *Proc. 25th Annual ACM Symp. Theory of Computing*, Victoria, 1993.
- [13] S. Even, and G. Granot, *Rectilinear Planar Drawings with Few Bends in Each Edge*, Manuscript, Faculty of Comp. Science, the Technion, Haifa (Israel), 1993.
- [14] Even, S., and R.E. Tarjan, Computing an st -numbering, *Theoret. Comp. Science* 2 (1976), pp. 436–441.
- [15] Fáry, I., On straight lines representations of planar graphs, *Acta Sci. Math. Szeged*, 11 (1948), pp. 229–233.
- [16] Formann, M., T. Hagerup, J. Haralambides, M. Kaufmann, F.T. Leighton, A. Simvonis, E. Welzl and G. Woeginger, Drawing graphs in the plane with high resolution, *Proc. 31th Ann. IEEE Symp. on Found. of Comp. Science*, St. Louis, 1990, pp. 86–95.
- [17] Fraysseix, H. de, J. Pach and R. Pollack, How to draw a planar graph on a grid, *Combinatorica* 10 (1990), pp. 41–51

that a canonical ordering or a 4-connected triangular planar graph is possible, in which every vertex v_k has $out(v_k) \geq 2$, which decreases the width of a visibility representation by a factor 2. Using this result, Kant [27] proved that a visibility representation of a general planar graph can be constructed on a grid of size at most $(\lfloor \frac{3}{2}n \rfloor - 3) \times (n - 1)$. Independently, Nummenmaa also showed a linear time algorithm for constructing a visibility representation, using the canonical ordering of a triangulated planar graph, also requiring an $(2n - 5) \times (n - 1)$ grid.

We like to finish the paper with mentioning several open problems and directions for further research.

- Decrease some of the bounds with respect to the grid size, number of bends or minimum angle, given in this paper. Recently, Chrobak & Kant improved the grid bound for convex drawing of triconnected planar graphs to $(n - 1) \times (n - 1)$.
- Is it possible to use the *lmc*-ordering in other drawing representations as well, to obtain better results in planar graph drawings on a grid?
- Inspect the more combinatorial aspects of the *lmc*-ordering. It is not very hard to prove that an *lmc*-ordering for G also directly defines an *lmc*-ordering of the dual graph, but maybe other combinatorial observations can be made..
- Can an arbitrary planar graph be drawn planar with straight lines such that the minimum angle is $> \Omega(\frac{1}{d})$? (See also [31] and [12] for this question.)
- Devise a dynamic algorithms for the sequential algorithms of this paper. In [6] a dynamic framework for graph drawing problems is described, but this approach seems not to work here. Very recently, He & Kao presented a parallel implementation for finding a canonical ordering, and computing the convex drawing in $\mathcal{O}(\log^4 n)$ time, requiring $\mathcal{O}(n^2)$ processors.

Acknowledgements

The author wishes to thank Hans Bodlaender, Jan van Leeuwen and Therese Biedl, for giving a lot of helpful comments.

References

- [1] Bhasker, J., and S. Sahni, A linear algorithm to find a rectangular dual of a planar triangulated graph, *Algorithmica* 3 (1988), pp. 147–178.
- [2] Biedl, T., and G. Kant, *A Better Heuristic for Planar Orthogonal Drawings*, RUTCOR Research Report (to appear), 1994.
- [3] Booth, K.S., and G.S. Lueker, Testing for the consecutive ones property, interval graphs and graph planarity testing using PQ-tree algorithms, *J. of Computer and System Sciences* 13 (1976), pp. 335–379.

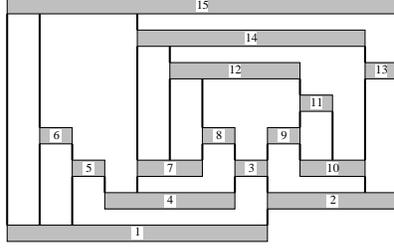


Figure 16: The visibility representation of the graph of Figure 2.

Payne. They build a tree on the vertices, where the edges contain the amount of shift for every edge. When applying this technique for our canonical ordering, there is no reason to compute a *leftmost* canonical ordering. In Chrobak & Kant [8], this is described in more detail.

Since every *lmc*-ordering is also an *st*-ordering, we can use the *lmc*-ordering in various drawing applications, where the *st*-ordering is used. We now focus the attention on the construction of a visibility representation by the *lmc*-ordering. In a visibility representation every vertex is mapped to a horizontal segment, and every edge is mapped to a vertical line, only touching the two vertex segments of its endpoints. In figure 16 an example is given. The visibility representation is interesting of its practical consequences. The interesting fact of using the *lmc*-ordering is that now we have the shift-values on the edges instead of on the vertices, and the algorithm becomes quite simple and may lead to more compact representations.

Hereto, when adding V_k , the y -coordinates of the vertices, already placed, are interesting, as well as the x -coordinates of (c_l, v_k) and (c_r, v_k) , where c_l and c_r are the left- and rightvertex of V_k , resp. Therefore we associate an y -coordinate, $y(v)$ to every vertex, and an x -coordinate, $x(u, v)$, and a *shift*-variable, $shift(u, v)$, to every edge (u, v) .

The horizontal segment, representing a vertex v must have length at least $out(v) - 1$. If we add $V_k = \{z\}$, then the horizontal segment, representing z goes from $(x(z, c_l), y(z))$ to $(x(z, c_r), y(z))$. If $V_k = \{z_1, \dots, z_\ell\}$, $\ell > 1$, then we can draw the horizontal segments of z_1, \dots, z_ℓ alternately on some height Y and $Y + 1$, as shown in Figure 16. The implementation of the complete algorithm follows now directly and is left to the reader.

Theorem 7.1 *There is a linear time algorithm to construct a visibility representation of a planar graph on a grid of size at most $(2n - 5) \times (n - 1)$.*

Notice that Kant proved [26] proved that every graph G can be Notice that the graph, consisting of nested triangles, indeed requires a grid of size $(2n - 5) \times (n - 1)$. If for every vertex v_i , $out(v_i) \geq 2$ holds, then this leads to a visibility representation on a grid of size at most $(n - 1) \times (n - 1)$. Indeed, in [28] Kant & He proved

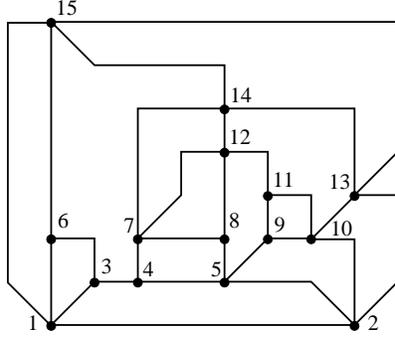


Figure 15: Drawing the graph of Figure 2 with bends.

when $y(c_l) \geq y(c_r)$.) In each edge, one bend was already assigned to the insertion step of c_l and c_r , hence adding v_k with $in(v_k) = 2$ requires at most one bend for the incoming edges. If $in(v_k) \geq 3$, then at most $2 \cdot in(v_k) - 4$ extra bends are required for the incoming edges. Edge (v_1, v_2) requires no bends. Counting this leads to totally at most $5n - 15$ bends. Every edge goes at most once vertical and once horizontal, hence requiring 3 bends in worst-case and by Lemma 6.2, has length $\mathcal{O}(n)$. \square

Theorem 6.4 *There is a linear time and space algorithm to draw a triconnected d -planar graph planar on an $(2n - 6) \times (3n - 9)$ grid with at most $5n - 15$ bends and minimum angle $> \frac{1}{d-2}$, in which every edge has at most 3 bends and length $\mathcal{O}(n)$.*

In Figure 15 the drawing of the graph of Figure 2 is given.

Notice that Kant [26] proved that every graph G can be augmented by adding edges to a triconnected planar graph G' such that $\Delta(G') \leq \lceil \frac{3}{2}\Delta(G) \rceil + 3$. This yields the following theorem.

Theorem 6.5 *There is a linear time and space algorithm to draw a planar graph planar on an $(2n - 6) \times (3n - 9)$ grid with at most $5n - 15$ bends and minimum angle $> \frac{4}{3\Delta(G)+1}$, in which every edge has at most 3 bends and length $\mathcal{O}(n)$.*

7 Final remarks and open problems

In this paper, a new ordering of the vertices and faces of a triconnected planar graph is introduced. This ordering leads to various algorithms to draw a planar graph on a grid. In some cases considerable improvements on existing results are obtained, in other cases new bounds are achieved. All algorithms can be implemented by straightforward techniques to run in linear time and space. Instead of using the shift-method, explained in section 2, we could also use the technique of Chrobak &

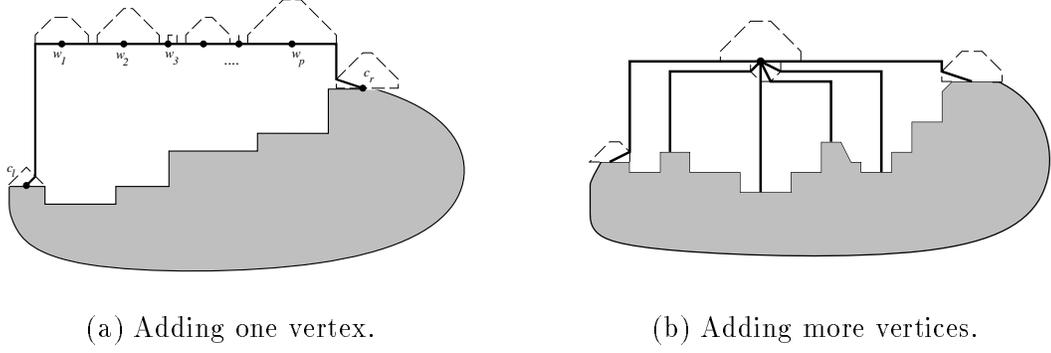


Figure 14: Adding vertices in the mixed model.

easily proved, hence assume $\deg(v) \geq 6$. The size of the angle is $\arctan(\frac{1}{\text{out}_r(v)}) \geq \arctan(\lfloor \frac{2}{\deg(v)-2} \rfloor)$. Using the potence series of the arctan we know that for $|x| < 1$, $\arctan(x) = x - \frac{1}{3}x^3 + \frac{1}{5}x^5 - \frac{1}{7}x^7 + \dots \geq x - \frac{1}{3}x^3$. Since $\lfloor \frac{2}{\deg(v)-2} \rfloor \geq \frac{2}{\deg(v)-1}$, we obtain that $\alpha \geq \frac{2}{\deg(v)-1} - \frac{1}{3}(\frac{2}{\deg(v)-1})^3 \geq \frac{2}{\deg(v)}$, which completes the proof. \square

Lemma 6.2 *The gridsize is at most $(2n - 6) \times (3n - 6)$.*

Proof: For the width notice that $x(c_{i+1}) = x(c_i) + \text{out}_r(c_i) + \text{out}_l(c_{i+1})$ holds in every step on the outerface if $y(c_{i+1}) \neq y(c_i)$ and $x(c_{i+1}) = x(c_i) + \text{out}_r(c_i) + \text{out}_l(c_{i+1})$ otherwise. If $y(c_{i+1}) = y(c_i)$ then (c_i, c_{i+1}) is not an outgoing edge of any vertex. Let us call (c_i, c_{i+1}) in this case *unmarked*. Counting leads to a horizontal distance of at most $\sum_{1 \leq i < n} \text{out}_l(v_i) + \text{out}_r(v_i) + \text{number of unmarked edges} = \sum_{1 \leq i < n} (\text{out}(v_i) - 1) = 2n - 6$.

Adding a vertex v_k requires more increase in height per vertex than adding a face, hence assume we add a vertex v_k in every step. Let the incoming edges of vertex v_k come from vertices u_1, \dots, u_r , then $y(v_k) \geq \max_{1 \leq i \leq r} \{y(u_i)\} + \max\{1 + \text{in}_r(v), \text{out}_r(u_1), \text{out}_r(u_r)\}$. The increase for every vertex v_k during the insertions is at most $1 + \text{in}_r(v_k) + \text{out}_r(v_k)$. Summarizing this for all vertices leads to a total distance in Y -direction of at most $3n - 6$ units. \square

Lemma 6.3 *There are at most $5n - 15$ bends. Every edge has at most 3 bends and length $\mathcal{O}(n)$.*

Proof: All outgoing edges of vertex v , except the one going straight upwards, requires one bend in worst-case to go in vertical direction. We assign these bends to the insertion step of v . Adding a face requires less bends per vertex than adding a vertex, so assume we only add vertices v_k . If $\text{in}(v_k) = 2$ and $y(c_l) \geq y(c_r)$, then there will come at most 1 bend in (c_l, v_k) and 2 bends in (c_r, v_k) . (A similar holds

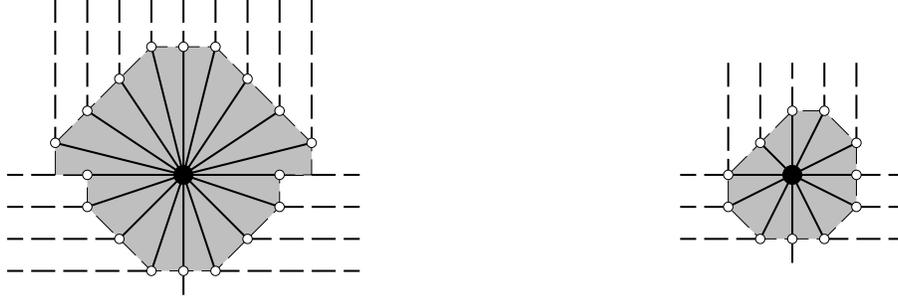


Figure 13: Examples of bounding boxes.

Adding a vertex v

For the y -direction, we simply set $y(v) = \max\{y(c_l) + out_l(c_l) + 1, y(c_r) + out_l(c_r) + 1\}$. In x -direction, the problem is a little more difficult: let $u_1, \dots, u_{in(v)}(v)$ be neighbors of v , corresponding with the left-to-right order of incoming edges of v . Let $u = u_{in_l(v)}$. then we want to have $x(v) = x(u)$, but also (for the outgoing edges), we want to have $x(v) > x(c_l) + out_r(c_l) + out_l(v)$. Hence we set $x(v) = \max\{x(u), x(c_l) + out_r(c_l) + out_l(v)\}$, and shift $u_{in_l(v)}, \dots, u_{in(v)-1}$ to the right (if $x(c_l) + out_r(c_l) + out_l(v) > x(u)$). For the rightvertex c_r we set $x(c_r) = \max\{x(c_r), x(v) + out_r(v) + out_l(c_r) + 1\}$.

Adding $z_1, \dots, z_\ell, \ell > 1$.

Now every vertex z_1, \dots, z_ℓ has precisely two incoming edges, hence they are placed on a horizontal line at height $\max\{y(c_l) + out_l(c_l), y(c_r) + out_l(c_r)\} + 1$. In the x -direction we don't have to deal with the incoming edges, hence we set $x(z_1) = x(c_l) + out_r(c_l) + out_l(z_1) + 1$, we set $x(z_i) = x(z_{i-1}) + out_r(z_{i-1}) + out_l(z_i) + 1$ ($1 < i \leq p$), and we set $x(c_r) = \max\{x(c_r), x(z_\ell) + out_r(z_\ell) + out_l(c_r) + 1\}$. Figure 14 makes this more precise.

Analysis of the algorithm

Using the shift-technique, explained in section 2, it is not difficult to obtain a linear time and space algorithm, satisfying the constraints with respect to width and height, as given in the two relevant steps. Therefore we now consider in detail the number of bends, the size of the minimum angle and the total grid size.

Lemma 6.1 *The size of the minimum angle is $\frac{2}{\Delta}$, where Δ is the maximum degree of G .*

Proof: Let v have maximum degree. The minimum angle, say α , is reached at an outpoint which is neighbored to a horizontal line. If $deg(v) < 6$, then it is

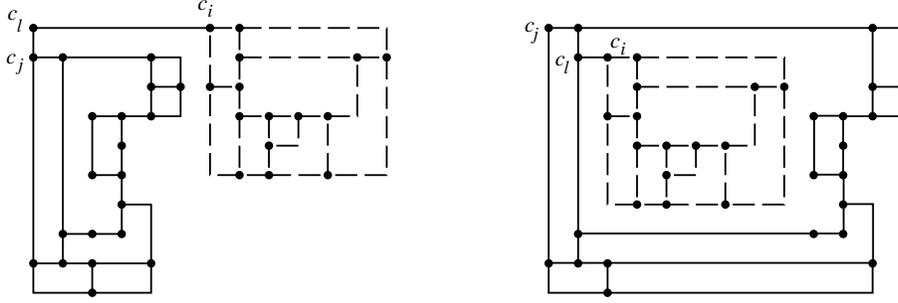


Figure 12: Orthogonal drawing of the blocks.

6 The mixed model

In this section we use the drawing framework, introduced in Section 3.1, to draw any triconnected d -planar graph G on an $(2n - 6) \times (3n - 9)$ grid such that there are at most $5n - 15$ bends and the minimum angle is at least $\frac{2}{d}$ radians. All vertices and bend coordinates will be placed on grid points only. Every edge will have at most three bends and length $\mathcal{O}(n)$.

Every edge, say (u, v) will have the following format. From u it goes to an *outpoint* of u , say b_o , from b_o it goes in vertical direction to a point, say b' , from b' it goes in horizontal direction to an *inpoint*, say b_i of v , and from b_i it goes to v . Important question is here what the coordinates of the in- and outpoints are, and how to compute them.

Let $out_l(v) = \lfloor \frac{out(v)-1}{2} \rfloor$ and $out_r(v) = \lceil \frac{out(v)-1}{2} \rceil$. Similar for $in(v)$, i.e., $in_l(v) = \lfloor \frac{in(v)-3}{2} \rfloor$ and $in_r(v) = \lceil \frac{in(v)-3}{2} \rceil$. The idea is to place the outpoints of v on the following places: the diagonal lines from $(x(v) - out_l(v), 1)$ to $(x(v) - 1, y(v) + out_l(v))$, and from $(x(v) + out_r(v), 1)$ to $(x(v) + 1, y(v) + out_r(v))$, and using $(x(v), y(v) + out_l(v))$. For the inpoints it is defines similarly: the diagonal lines from $(x(v) - in_l(v), -1)$ to $(x(v) - 1, y(v) - out_l(v))$, and from $(x(v) + out_r(v), -1)$ to $(x(v) + 1, y(v) - out_r(v))$, and the three points $(x(v) - out_l(v), y(v))$, $(x(v), y(v) - out_l(v))$ and $(x(v) + out_r(v), y(v))$. See Figure 13 for some examples, and the corresponding bounding boxes.

The width of the bounding box is $\max\{out(v) - 1, in(v) - 3\}$, the height is $in_l(v) + out_l(v)$. The idea is to insert the vertices of the canonical ordering such that the bounding boxes do not intersect or touch. Also outgoing edges may not cross or overlap, i.e., this means that for every pair of consecutive vertices c_i, c_{i+1} on the outface, $x(c_{i+1}) > x(c_i) + out_r(c_i) + out_l(c_{i+1})$ must hold. We can now explain the different adding steps as follows:

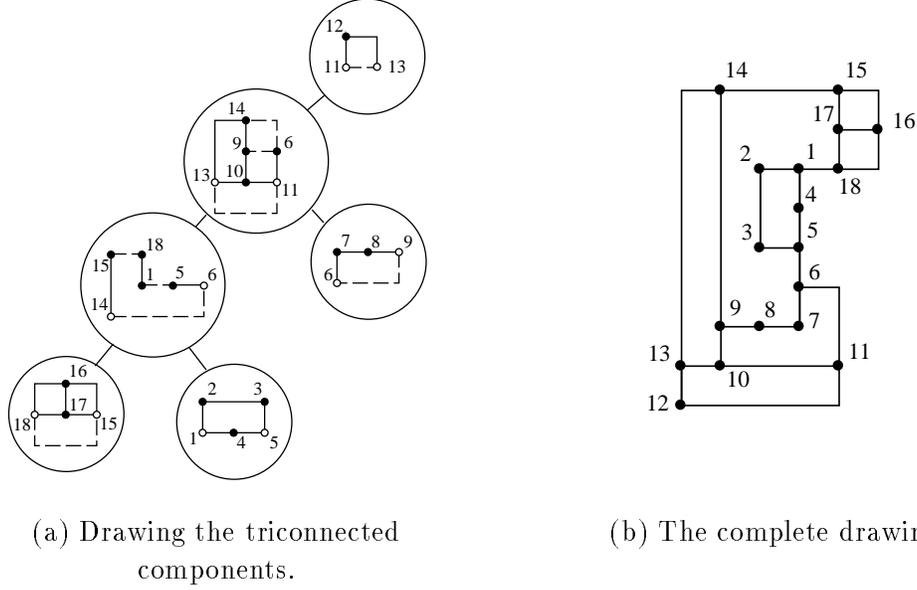


Figure 11: Orthogonal drawing of a biconnected 3-planar graph.

then T contains an S-node, and we assumed that b_r is an S-node. One easily observes that no bends are introduced, when we consider an S- or S'-node. Hence the following theorem is obtained:

Theorem 5.8 *There is a linear time and space algorithm to draw a biconnected 3-planar graph on a grid of size at most $\lfloor \frac{n}{2} \rfloor \times \lfloor \frac{n}{2} \rfloor$, with at most $\lfloor \frac{n}{2} \rfloor + 1$ bends, with the property that there is a spanning tree of $n - 1$ straight-line edges, while all non-tree edges have at most 1 bend (if $n > 4$).*

In Figure 11 the orthogonal drawing of the graph in Figure 9 is given.

We extend the algorithm 3-ORTHOGONAL to draw arbitrary 3-planar graphs orthogonally. Assume all biconnected components B_i of G are drawn orthogonally, with the cutvertex, say v_i , in the upperleft corner. (Since v_i has degree 2 in B_i and the root of the corresponding SPQR-tree was assume to be an S-node, this can easily be obtained.) We draw the blocks recursively, and merge it into one drawing, as shown in Figure 12.

No extra bends are introduced in the drawing. Also the required area for drawing blocks B_i and B_j is at most $\lfloor \frac{|B_i|+|B_j|}{2} \rfloor \times \lfloor \frac{|B_i|+|B_j|}{2} \rfloor$, which completes the following theorem:

Theorem 5.9 *There is a linear time and space algorithm to draw any 3-planar graph on an $\lfloor \frac{n}{2} \rfloor \times \lfloor \frac{n}{2} \rfloor$ grid with at most $\lfloor \frac{n}{2} \rfloor + 1$ bends, with the property that there is a spanning tree of $n - 1$ straight-line edges, while all non-tree edges have at most 1 bend (if $n > 4$).*

If b_i is an S' or R-node, then b_j is an S-node. We can draw B_j such that (s_i, t_i) is a straight line, and since B_i can be drawn such that s_i and t_i have the same y -coordinate, the problem is easily solved. Hence we now assume that b_j is an R-node, thus B_j is a triconnected graph and b_i is an S-node. We now place B_i inside B_j as shown by the templates in Figure 10. From these replacements, the following lemma is easy to prove:

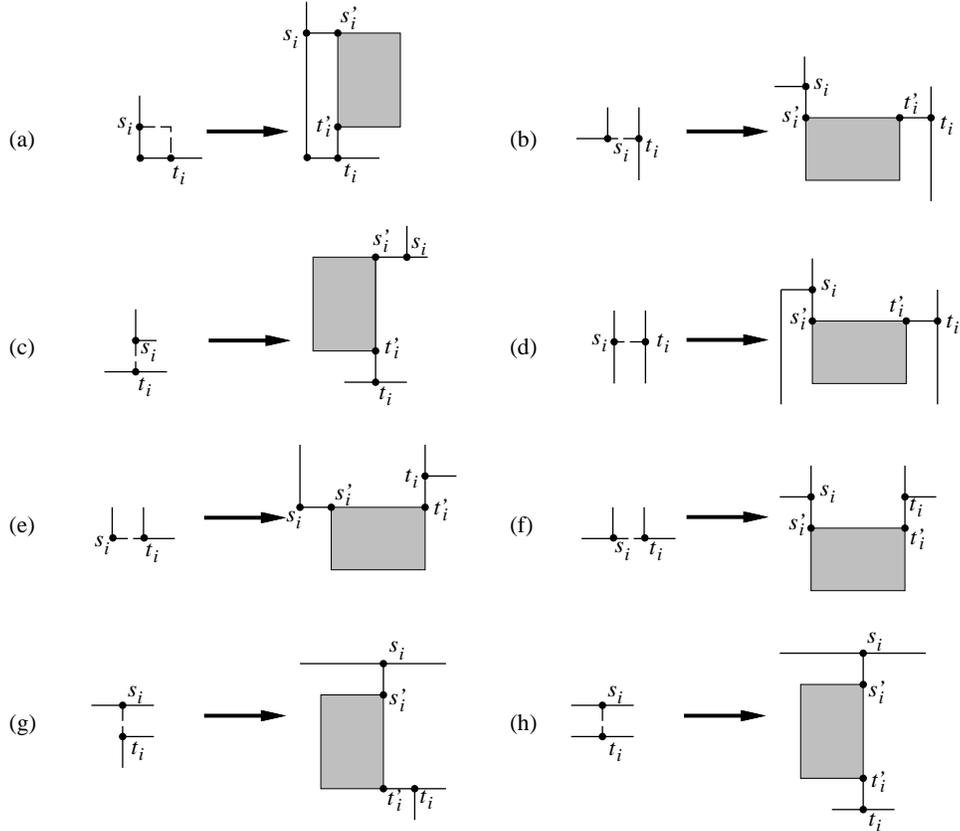


Figure 10: Replacing virtual edges by triconnected components.

Lemma 5.7 *After replacing a virtual edge (s_i, t_i) by the corresponding orthogonal drawing of B_i in an orthogonal drawing of B_j , the total required grid size is at most $\lfloor \frac{|B_i|+|B_j|}{2} \rfloor \times (\lfloor \frac{|B_i|+|B_j|}{2} \rfloor - 1)$.*

This means that after replacing all virtual edges of B_j by the triconnected components, we obtain an orthogonal drawing of size $\lfloor \frac{n'}{2} \rfloor \times (\lfloor \frac{n'}{2} \rfloor - 1)$, where n' is the number of nodes of the subgraph of G , corresponding to the subtree of T , rooted at b_j . We continue this approach until we are at root b_r of T . If G is not triconnected,

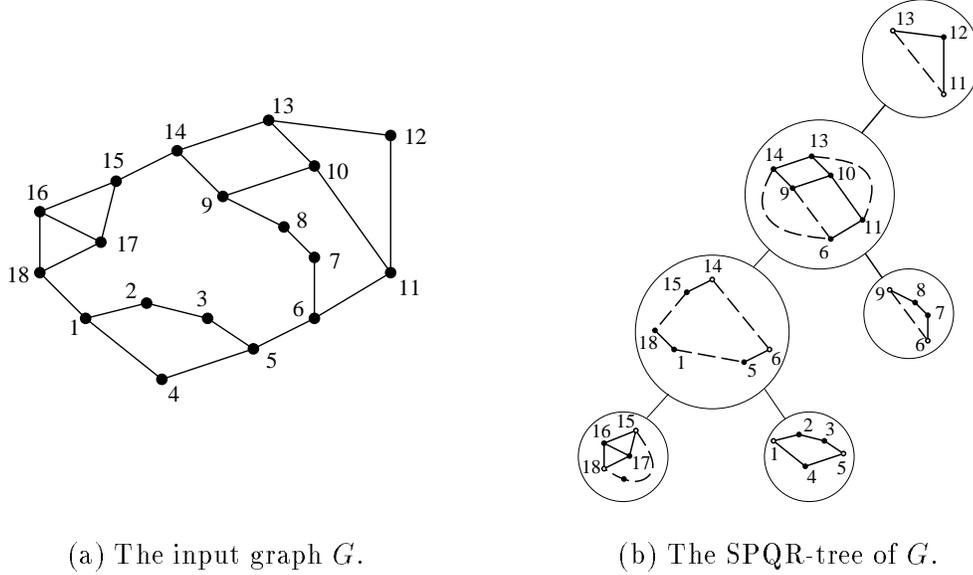


Figure 9: Example of a biconnected 3-planar graph and its SPQR-tree.

parent of an S'-node are S-nodes. The neighbors of an S-node and the children of an S'-node are R- or S'-nodes.

The algorithm for drawing a biconnected 3-planar graph G draws the triconnected components of G in order of visiting the corresponding nodes in the SPQR-tree bottom-up. Let s_i and t_i be the poles of a leaf b_i in T :

b_i is an R-node B_i is drawn by the algorithm 3-ORTHOGONAL with $v_1 = s_i$ and $v_2 = t_i$.

b_i is an S'-node We draw B_i as a rectangle such that s_i and t_i are the lowerleft and lowerright corner. If one path between s_i and t_i in B_i contains at least 2 other vertices, then vertices are placed in the other 2 corner points.

b_i is an S-node Let (s_i, s'_i) and $(t_i, t'_i) \in B_i$, $s'_i \neq t_i$ and $t'_i \neq s_i$. Let B'_i be $B_i - \{s_i, t_i\}$. Draw B'_i as a rectangle with s'_i as lowerleft, t'_i as a right corner, and other vertices of B'_i (if any) placed in the other corners. We draw s_i one below s'_i and t_i one right of t'_i .

Let $|B_i| = n'$. It follows that the used area of the described drawings is at most $\lfloor \frac{n'}{2} \rfloor \times (\lfloor \frac{n'}{2} \rfloor - 1)$. Let $x(b_i) = x(t_i) - x(s_i)$ and $y(b_i) = y(t_i) - y(s_i)$.

The idea now is to draw a triconnected component B_j , after all descendants of b_j in T are already drawn. Let b_i be a child of b_j . We want to stretch the drawing of B_j such that the edge (s_i, t_i) has width $x(b_i)$ and height $y(b_i)$, and that B_i can be placed inside without crossing edges.

are drawn similar as before. Notice that using this strategy, every triconnected planar graph G with n vertices can be drawn orthogonally on a grid of size at most $\frac{n}{2} \times (\frac{n}{2} - 1)$, with at most $\frac{n}{2} + 1$ bends (n is even), in which there is a spanning tree, using only straight-line horizontal and vertical edges. All non-tree edges have at most one bend (if $n > 4$).

We notice that better bounds can be obtained if the dual graph G^* of G is a 4-connected planar graph in which all internal faces are triangles. It has been shown by Bhasker & Sahni [1] that in this case G can be drawn orthogonally in linear time such that there are at most 4 bends.

5.3 General 3-Planar Graphs

To apply the algorithm to general connected 3-planar graphs, we have to split up the graph into its biconnected and triconnected components.

The triconnected components of a biconnected graph G are defined as follows: If G is triconnected itself it is the unique triconnected component. Otherwise let (u, v) be a separation pair of G . We partition G into two subgraphs G_1 and G_2 which have only vertices u and v in common. We continue the decomposition process recursively on $G'_1 = G_1 + (u, v)$ and $G'_2 = G_2 + (u, v)$ until no decomposition is possible. The added edges are called virtual edges. The resulting graphs are each either a triconnected simple graph, or a set of three multiple edges (triple bond), or a cycle of length 3 (triangle). The triconnected components of G are obtained from such graphs by merging the triple bonds into maximal sets of multiple edges (bonds), and the triangles into maximal simple cycles (polygons). The triconnected components are unique.

Next we need the *SPQR-tree*, a versatile data structure that represents the decomposition of a biconnected graph into its triconnected components [10]. The SPQR-tree T is defined as follows: for every triconnected component we create an R-node, for every polygon an S-node, for every bond a P-node, and for every edge a Q-node. The edges in T are defined as follows: Let u, v be nodes in T . If u is a Q-node then there is an edge between u and v if the edge represented by u belongs to the triconnected component represented by v . Otherwise there is an edge between u and v if and only if they contain the same virtual edge added in the same step of the decomposition process. In Figure 9 an example is given of a biconnected graph and the corresponding SPQR-tree.

Let the triconnected component of node b_i in T be defined by B_i , and let s_i and t_i be the two vertices of B_i , called the *poles*, also shared by the triconnected component of $parent(b_i)$. Since the maximum degree of G is 3, it follows that the neighbors of a P-node are not R-nodes. Root T at an arbitrary S-node b_r , and let b_{i_1} and b_{i_2} be the two children of an P-node b_i in T . We merge B_{i_1} and B_{i_2} as described above, yielding a cycle. We represent the resulting cycle by an S'-node. In this way we remove all P-nodes from T . We also remove all leaves (which are Q-nodes) from T . Notice that since the maximum degree is 3, the neighbors of an R-node and the

assume we added z_1, \dots, z_ℓ from c_l to c_r ;
for $i := 1$ **to** ℓ **do** $x(z_i) := x_{insert}(z_i) + \sum_{1 \leq j < i} shift(z_j)$ **rof**;
if $\ell = 1$ **then** $shift(c_r) := shift(z_1)$ **else** $shift(c_r) := x(z_\ell) - x_{insert}(c_j)$
rof;
END 3-ORTHOGONAL

Lemma 5.5 *The number of bends is at most $\frac{n}{2} + 2$.*

Proof: Since $m = \frac{3}{2}n$, we add at most $\frac{n}{2} - 2$ times a vertex v with $in(v) = 2$, each one introduces one bend. The edge (v_1, v_2) introduces 2 bends, as well as adding v_n . \square

Lemma 5.6 *The gridsize is at most $\frac{n}{2} \times \frac{n}{2}$.*

Proof: Edge (v_1, v_2) gives 1 unit in X - and Y -direction. Then we add $\frac{n}{2} - 1$ times a face with $\ell \geq 1$ vertices, increasing the X -direction with at most $\ell - 1$ units and the Y -direction (except the first time) by 1 unit. Adding v_n increases the Y -direction by 1 unit. Counting this together leads to at most $\frac{n}{2}$ units in X -direction and $\frac{n}{2}$ units in Y -direction. \square

In Figure 8(b), an example is given of a triconnected 3-planar graph.

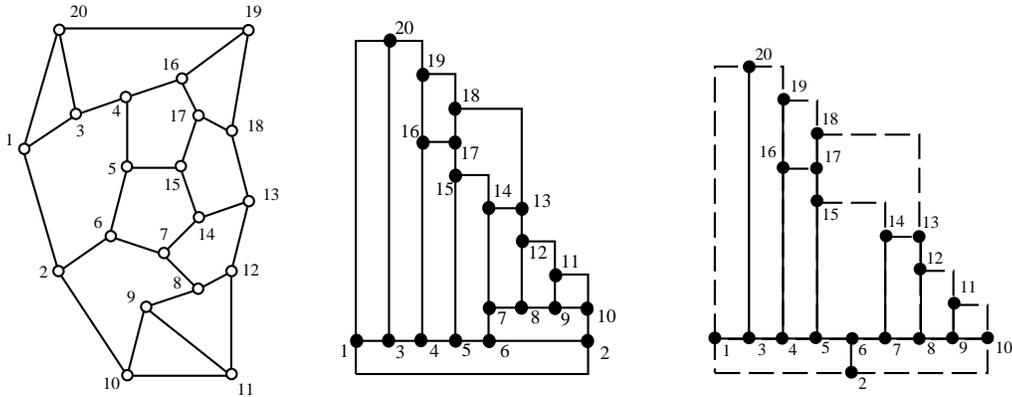
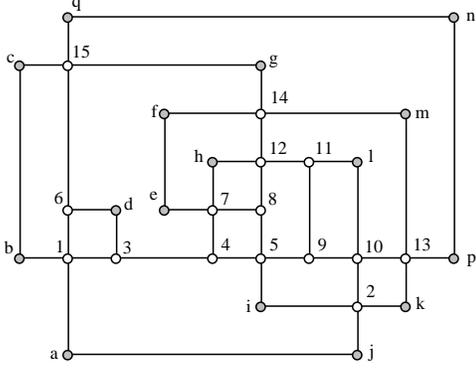
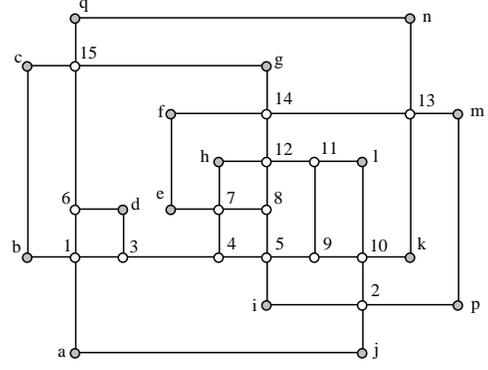


Figure 8: Orthogonal drawing of a triconnected 3-planar graph.

We can change the drawing as follows, such that there is one bend less, and there is a spanning tree, using only straight-line edges (if $n > 4$). Let the vertices of the first drawn face be numbered $v_1, v_i, v_{i-1}, \dots, v_3, v_2$. We place $v_1, v_i, v_{i-1}, \dots, v_3$ on a horizontal line, and place v_2 on $(x(v_3), y(v_3) - 1)$. Let F' be the other face, where (v_2, v_3) belongs to. Let v_j, \dots, v_k be the other vertices of F' . We draw v_j, \dots, v_k on a horizontal line on height $y(v_3)$, as shown in Figure 8(c). The remaining faces



(c) Orthogonal drawing.



(d) At most 2 bends in every edge.

Figure 7: Orthogonal drawing of the 4-planar graph of Figure 2.

Assume first that G is triconnected. By Euler's formula, n is even, $m = \frac{3}{2}n$ and $f = \frac{n}{2} + 2$. Let an lmc -ordering of G be given. Similarly as in Section 5.1 there are four directions to connect an edge at v , namely, $left(v)$, $up(v)$, $right(v)$ and $down(v)$. Every vertex v (except v_1, v_2 and v_n) has one outgoing edge, and we connect this edge via $up(v)$ at v . We start with placing v_1 and v_2 at $(0, 1)$ and $(1, 1)$. edge (v_1, v_2) goes via $down(v_1)$ and $down(v_2)$, hence via $(0, 0)$ and $(1, 0)$. In step 2, the vertices z_1, \dots, z_ℓ of V_2 are placed on the horizontal line between v_1 and v_2 , i.e., via $right(v_1)$ and $left(v_2)$. In every step k , $3 \leq k < K$, we place z_1, \dots, z_ℓ also on a horizontal line of height $1 + \max\{y(c_l), y(c_r)\}$, with c_l and c_r the left- and rightvertex of V_k . If $\ell > 1$ then we shift the drawing such that $x(z_1) = x(c_l)$ and $x(z_\ell) = x(c_r)$. Since $in(V_k) = 2$ for $2 \leq k < K$ and $in(v_n) = 3$, it follows that $K = f$, with f the number of faces in G . Notice that $f = \frac{n}{2} + 2$ (n is even). The complete algorithm can now be described as follows:

3-ORTHOGONAL

$P(v_1) := (0, 1); P(v_2) := (1, 1);$

for $k := 3$ **to** $f - 1$ **do**

 assume we add z_1, \dots, z_ℓ ($\ell \geq 1$), from c_l to c_r ;

$y(z_1) := \dots := y(z_\ell) := 1 + \max\{y(c_l), y(c_r)\}$;

 update $x(c_l)$ and $shift(c_r)$;

$x(z_1) := x(c_l)$;

for $l := 2$ **to** $\ell - 1$ **do** $x(z_l) := x(z_1) + l - 1$ **rof**;

if $\ell > 1$ **then** $x(z_\ell) := \max\{x(z_1) + \ell - 1, x(c_r) + shift(c_r)\}$;

$shift(c_r) := \max\{shift(c_j), x(z_\ell) - x(c_j)\}$

rof;

$P(v_n) := (x(c_l), 1 + \max\{y(c_l), y(c_l), y(c_r)\})$, where $in(v_n) = \{c_l, c_l, c_r\}$;

for $k := f$ **downto** 2 **do**

step	# vertices	increase in width
1, adding (v_1, v_2)	2	3
$k, \ell = 1$ and $in(z_1) = 2$	1	1
$k, \ell = 1$ and $in(z_1) = 3$	1	0
$k, \ell > 1$	ℓ	ℓ
K , adding v_n	1	0

This leads to a total width of at most n . □

Indeed, in our solution, the edge, using $up(v_n)$ has at most three bends, all other edges have at most two bends. How can we avoid the edge with three bends? Indeed, Even & Granot proved that any orthogonal drawing of the 4-planar triangulated planar graph on 6 vertices (octahedron) requires at least one edge with at least three bends [13]. In our case, if there is a vertex v with $deg(v) = 3$ then we can set $v_n = v$. Otherwise let $n > 6$. Then there is a face with at least 4 vertices, which we choose to be the outerface. Let v_{n_1}, \dots, v_{n_4} be the neighbors of v_n from left to right. If edge (v_n, v_{n_4}) uses $right(v_{n_4})$, then we change the four directions of v_{n_4} such that $up(v_{n_4})$ is used for (v_{n_4}, v_n) . Since $mark(v_{n_4}) = right$ (by definition), it follows that at most one extra bend is introduced. Moreover, since $n_4 \neq 2$ it follows that all other edges still have at most 2 bends. This completes the following theorem.

Theorem 5.4 *There is a linear time and space algorithm to draw every triconnected 4-planar graph G orthogonally on an $n \times n$ grid with at most $\lceil \frac{3}{2}n \rceil + 4$ bends, such that every edge has at most two bends and length $\mathcal{O}(n)$ if $n > 6$.*

In Figure 7 the orthogonal drawing of the graph of Figure 2 is given. In particular, in Figure 7(d) it is shown how to change v_k such that all edges have at most 2 bends. The previous bound on the number of bends was $2n + 4$, given by Tamassia & Tollis [37]. Hence our algorithm improves this result considerably for triconnected planar graphs. In the algorithm of Tamassia & Tollis, every edge gets at most 4 bends, hence we also improve this bound. Very recently, Biedl & Kant presented a linear time algorithm for constructing an orthogonal representation of a connected planar graph on an $n \times n$ grid, having at most $2n + 2$ bends, and every edge is bent at most twice. Notice that at most 2 bends in every edge is best possible, because if the planar graph contains a separating triangle on the vertices v_i, v_j, v_k , then at least one edge of the separating triangle has at least 2 bends in any orthogonal drawing.

5.2 Triconnected 3-Planar Graphs

In this section we present a linear time and space algorithm to draw every 3-planar graph with at most $\lfloor \frac{n}{2} \rfloor + 1$ bends on an $\lfloor \frac{n}{2} \rfloor \times \lfloor \frac{n}{2} \rfloor$ grid. This improves all previous bounds (from [35, 37]) and matches the worst-case lower bounds and, hence, is best possible. An interesting side-effect is that there is a spanning tree using $n - 1$ straight-line edges. All $m - n + 1 \leq \lfloor \frac{n}{2} \rfloor + 1$ non-tree edges have at most one bend.

If $\ell = 1$ then we have at most two bends if $in(z_1) = 3$, so assume $in(z_1) = 2$. The incoming edge, using $down(z_1)$, gets no extra bends, the other edge gets one more bend. If $right(c_l)$ and $up(c_r)$ are both free, then both edges, (c_l, z_1) and (c_r, z_1) are straight lines (using $left(z_1)$ and $down(z_1)$, resp). However, if $mark(z_1) = left$, then an extra bend is required for the outgoing edge of z_1 , using $right(z_1)$. We assign this extra bend to step k . Similar when $mark(z_1) = right$, hence in all cases at most one extra bend is introduced when $in(z_1) = 2$.

Also a similar assignment follows for edge (v_1, v_2) : we assign the extra bends of edges, using the connection $left(v_1)$ or $right(v_2)$, to step 1. Summarizing this leads to the following table:

step	# vertices	# edges	# bends
$\ell = 1, in(z_1) = 2$	1	2	1
$\ell = 1, in(z_1) = 3$	1	3	2
z_1, \dots, z_ℓ	ℓ	$\ell + 1$	ℓ

Every vertex v has $deg(v) \leq 4$, thus $m \leq 2n$. Consider the steps $2, \dots, K - 1$ in which $n - 3$ vertices and at most $2n - 5$ edges are added. Adding $V_k = \{z\}$ with $in(z) = 3$ occurs at most $\lceil \frac{n}{2} \rceil - 2$ times, because then at most $\lceil \frac{3n}{2} \rceil - 6$ edges are added, and at most $\lfloor \frac{n}{2} \rfloor + 1$ edges are added by at most $\lfloor \frac{n}{2} \rfloor - 1$ vertices. This yields at most $2(\lceil \frac{n}{2} \rceil - 2) + \lfloor \frac{n}{2} \rfloor - 1 + 8 = \lceil \frac{3n}{2} \rceil + 3$ bends. The edge, using $up(v_n)$ has at most three bends, all other edges have at most 2 bends. \square

Lemma 5.3 *The gridsize is at most $n \times n$.*

Proof: The increase in height in step k , $1 < k < K$, is at most ℓ , where $V_k = \{z_1, \dots, z_\ell\}$. In step 1 the increase in height is at most one, and in step K two, which proves the total height of n .

For the total width, we consider the different cases for step k , $1 < k < K$. Let $V_k = \{z_1, \dots, z_\ell\}$. If $\ell > 1$, then $x(z_i) - x(z_{i-1}) = 1$ ($1 < i < \ell$). If $x(z_\ell) - x(z_{\ell-1}) > 1$ then there is no increase in width at all in this step, since then $x(c_r) - x(c_l) > \ell - 1$. If (c_l, z_1) is horizontal, then this means an increase of one in width, if (c_l, z_1) is vertical and $mark(z_1) = right$, then one outgoing edge of z_1 has to go via $l(z_1)$, hence this means also an increase of one in width later. We assign this increase to step k . A similar holds for (z_ℓ, c_r) . Since $x(c_r) \geq x(c_l) + 1$ in step $k - 1$, it follows that the increase in width is at most ℓ in step k .

If $\ell = 1$ and $in(z_1) = 2$, then the width increases by one, due to the fact that an extra column might be necessary for the outgoing edge of z_1 via $left(z_1)$ when $mark(z_1) = right$ (similar when $mark(z_1) = left$). If $\ell = 1$ and $in(z_1) \geq 3$ then the width does not increase.

For (v_1, v_2) we assign the extra columns, required by the outgoing edges via $left(v_1)$ and $right(v_2)$ to step 1, yielding a starting width of three units. This gives the following table:

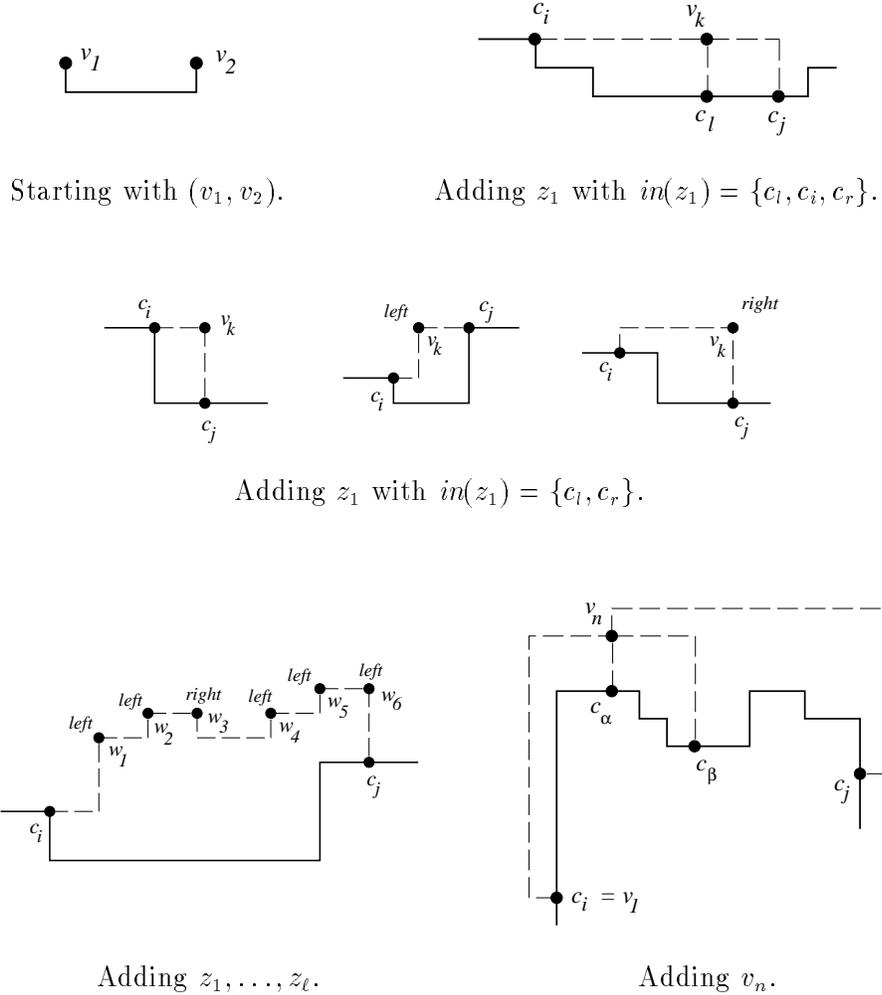


Figure 6: Adding vertices and faces to obtain an orthogonal drawing.

changing any of the x -values of vertices, already visited. The final x -coordinates are computed by traversing *Columns* and assigning ascending values to each element. Every vertex and bend then checks the value of the element it points to and stores it as its x -coordinate. This yields a planar orthogonal drawing.

Lemma 5.2 *The number of bends is at most $\lceil \frac{3}{2}n \rceil + 3$. One edge has at most three bends, all other edges have at most 2 bends.*

Proof: Let $V_k = \{z_1, \dots, z_\ell\}$. Assume first $\ell > 1$. For every vertex z_i , $down(z_i)$ and either $left(z_i)$ or $right(z_i)$ is used. The edge to $down(z_i)$ always requires one extra bend, the other ones not. If $mark(z_1) = right$, and $right(z_1)$ is used by (z_1, z_2) , then later an extra bend is required for the outgoing edge via $left(z_1)$. A similar holds for z_ℓ when $mark(z_\ell) = left$. Hence this implies at most ℓ bends.

is not free in G_k . Let c_l and c_r be the left- and rightvertex of v . We connect (c_l, v) at $right(c_l)$, if it is free, otherwise at $up(c_l)$, if it is free, otherwise at $left(c_l)$. The opposite direction is followed for c_r . We want to add v such that when $mark(v) = left$ then $left(v)$ is free after addition. Since v is the rightvertex of u_1 , we can use $left(v)$ for the edge (v, u_1) to the left. When $right(u_1)$ is used for (v, u_1) , then no bends occur in (v, u_1) , otherwise $up(u_1)$ is used, yielding one bend. Similar for $mark(v) = right$.

The algorithm, trying to achieve this as much as possible, can be described in a more elaborate way as follows:

```

4-ORTHOGONAL( $G$ );
  edge  $(v_1, v_2)$  via  $down(v_1)$  and  $down(v_2)$ ;
  for  $k := 3$  to  $K - 1$  do
    Let  $V_k = \{z_1, \dots, z_\ell\}$ ;
    • if  $\ell = 1$  and  $in(z_1) = \{c_l, c_i, c_r\}$  then
       $(c_l, z_1)$  via  $left(z_1)$ ;
       $(c_i, z_1)$  via  $down(z_1)$ ;
       $(c_r, z_1)$  via  $right(z_1)$ ;
    • if  $\ell = 1$  and  $in(z_1) = \{c_l, c_r\}$  then
      if  $mark(z_1) = left$  or  $(left(c_r)$  free and  $right(c_l)$  not free) then
         $(z_1, c_l)$  via  $down(z_1)$  and  $(z_1, c_r)$  via  $right(z_1)$ 
      else
         $(z_1, c_l)$  via  $left(z_1)$  and  $(z_1, c_r)$  via  $down(z_1)$ ;
    • otherwise ( $\ell > 1$ )
      if  $right(c_l)$  is free then  $(z_1, c_l)$  via  $left(z_1)$  else via  $down(z_1)$ ;
      for  $i := 2$  to  $\ell$  do
        if  $down(z_{i-1})$  is free then  $(z_{i-1}, z_i)$  via  $down(z_{i-1})$  else via  $right(z_{i-1})$ ;
        if  $mark(z_i) = left$  then  $(z_{i-1}, z_i)$  via  $down(z_i)$  else via  $left(z_i)$ ;
      rof;
      if  $left(c_r)$  is free then  $(z_{\ell-1}, z_\ell)$  via  $down(z_\ell)$ ;  $(z_\ell, c_j)$  via  $right(z_\ell)$  else
         $(z_{\ell-1}, z_\ell)$  via  $left(z_\ell)$ ;  $(z_\ell, c_r)$  via  $down(z_\ell)$ 
    rof;
  edges from  $c_l, c_{i_2}, c_{i_3}, c_r$  to  $v_n$  via  $left(v_n)$ ,  $down(v_n)$ ,  $right(v_n)$  and  $up(v_n)$ , resp.;
END 4-ORTHOGONAL

```

See Figure 6 for an illustration of the different cases.

There are several ways for computing the coordinates. Here we briefly describe the method, given by Biedl & Kant [2]: Remark that the y -coordinate of a vertex is never changed later, so we only have to worry about the x -coordinates. The crucial observation is that we need not know the values of the x -coordinates of incoming edges of v_i when adding v_i . We can do the following trick: throughout the algorithm maintain a list *Columns*. Every embedded vertex v contains a pointer $x(v)$ to one element of *Columns*. Whenever we want to add a column, we add a new element in *Columns*. By storing a list as a sequence of pointers we can do so without

5 Orthogonal Drawings

5.1 4-planar graphs

In this section we consider the problem of drawing a planar graph G on a rectilinear grid with orthogonal edges, i.e., the edges are polygonal chains of horizontal and vertical segments. The vertices are represented by points.

Theorem 5.1 *There are embedded triconnected 4-planar graphs G_n with $3n + 1$ vertices and $6n + 1$ edges, for which any layout requires at least $4n + 2$ bends.*

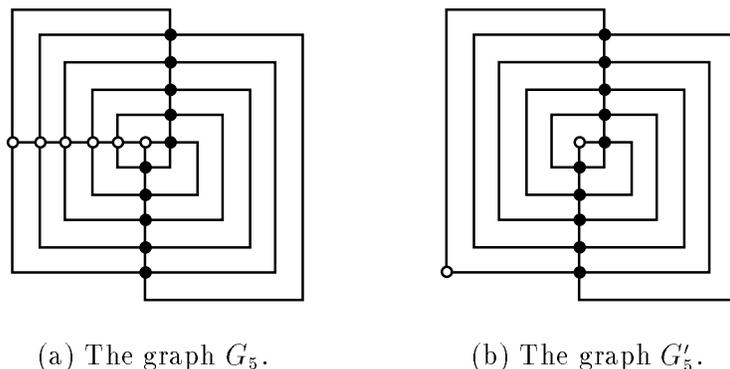


Figure 5: Lowerbound of $\frac{4}{3}(n - 1) + 2$ bends.

Proof: Consider the triconnected plane graph G_n with $3n + 1$ vertices, and its layout in Figure 5(a), which has $4n + 3$ bends. Notice that there are no bends in the edges between two white vertices. The vertices, which had degree 4 initially, have degree 2 now, and are deleted, while connecting the two incident edges. This leads to a biconnected planar graph G'_n with $2n + 2$ vertices (see Figure 5(b)). It is shown in corollary 4 in [39] that the shown layout in Figure 5(b) of G'_n is best possible with respect to the minimum number of bends, which is $4n + 2$. If there was a layout for G_n with fewer than $4n + 2$ bends, then there was a better layout of G'_n with fewer than $4n + 2$ bends, which contradicts Corollary 4 of [39]. \square

Let G be a triconnected 4-planar graph. Let an *lmc*-ordering of G be given. We introduce a variable $mark(v_i)$ for each vertex v , which is important when adding $V_k = \{v\}$ to G_{k-1} . v has at most two outgoing edges, say to u_1 and u_2 (from left to right). If v is the rightvertex of u_1 , and v is not the leftvertex of u_2 , then we set $mark(v) = left$ otherwise we set $mark(v) = right$.

There are four directions to connect an edge at v , namely, *left*, *right*, *up* and *down* of v . A direction is called *free* if there is no edge connected in that direction of v yet. The idea for the algorithm is as follows: we add v to G_{k-1} such that $down(v)$

$c_{\beta_1+1}, \dots, c_{i_2}$ are shifted to right by at least the value of the shift of $c_{i_1}, \dots, c_{\beta_1}$. This preserves the planarity in F_1 and completes the proof. \square

Finally we remove the added dummy edges from c_{i_j} to $c_{\beta_j}, c_{\beta_j+1}, \dots, c_{i_j-2}$ ($1 \leq j \leq s$).

Theorem 4.3 *There is a linear time and space algorithm to draw a triconnected planar graph convexly with straight-line edges on an $(2n - 4) \times (n - 2)$ grid.*

Our algorithm not only outperforms the algorithms of [42, 5], it is also much easier to implement than the algorithm of [5]. However, a drawback of the algorithm is that the drawings are not strictly convex, as those of [42, 5]. On the positive side, this algorithm gives a new proof that every triconnected planar graph admits a planar drawing, in which every interior face is convex. The outerface is a triangle. With respect to the tightness of the grid size we note that every strictly convex drawing of a cycle with n vertices requires an $\Theta(n^3)$ grid [30]. In Figure 4, the straight-line convex drawing of the graph in Figure 2 is given.

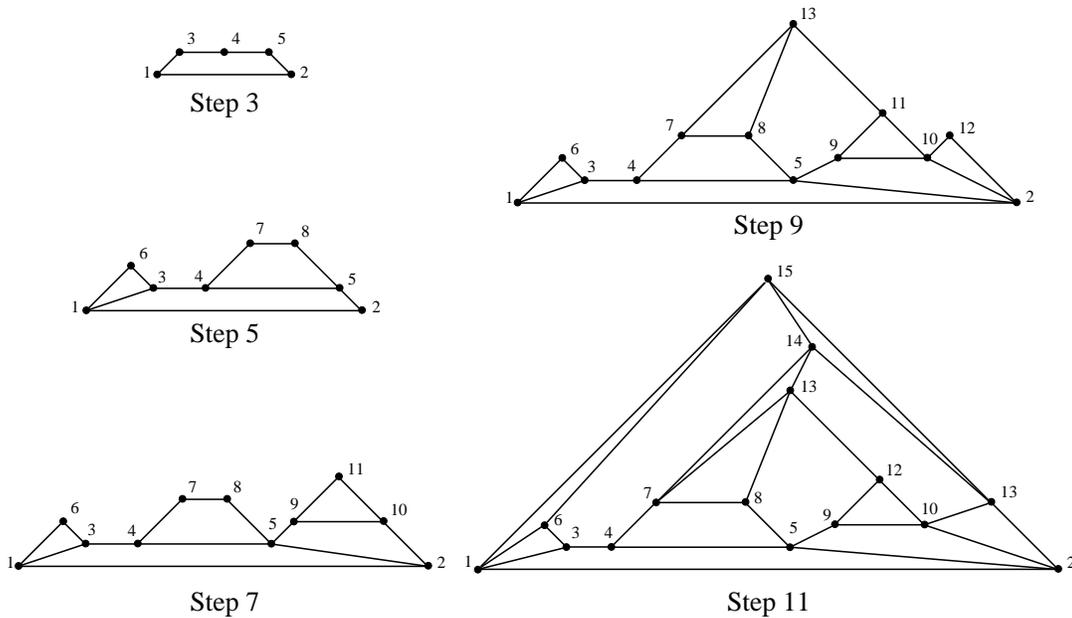


Figure 4: Convex drawing of the graph of Figure 2.

Proof: By definition of the *lmc*-ordering, every vertex $v \in V_k$ has a neighbor $w \in V_{k'}$, with $k' > k$. By definition of the algorithm `LINEARSTRAIGHT-LINEDRAW` it follows that $y(w) > y(v)$. In step k the vertices $c_{i_j+1}, \dots, c_{i_{j+1}-1}$ have already higher placed neighbors. Let c_{α_j} be the lowest placed vertex, with $i_j \leq \alpha_j < i_{j+1}$ and α_j minimal. If there is a vertex c_{β_j} with $y(c_{\beta_j}) = y(c_{\alpha_j})$, then $\beta_j = \alpha_j + 1$, because otherwise there would be a vertex c_γ , $\alpha_j < \gamma < \beta_j$, which does not have a higher placed neighbor. From c_{i_j} to c_{α_j} the vertices have strictly decreasing y -coordinate and from c_{α_j+1} to $c_{i_{j+1}}$ the vertices have strictly increasing y -coordinate. \square

Notice that all edges on C_{k-1} have slope $+1$, 0 and -1 before adding V_k . When adding z_1 , we shift $c_{\alpha_1}, \dots, c_{\beta_{s-1}}$ to right by one, and shift $c_{\beta_{s-1}+1}, \dots, c_{i_s}$ to right by two. As explained above, we draw z_1 at point $\mu(c_{i_1}, c_{i_s})$.

When $V_k = \{z_1, \dots, z_\ell\}$, we add only one face F_1 . Let B_1 be the path of C_{k-1} between c_l and c_r . B_1 also has the pattern of Lemma 4.1. We shift c_{α_1} and c_{β_1} to right by one, and $c_{\beta_1+1}, \dots, c_r$ to right by $2p$. z_1, \dots, z_ℓ are placed as explained above.

This yields the following slopes after adding V_k :

- The slope of edge $(c_{\alpha_1-1}, c_{\alpha_1})$ is in the range $[-1, 0)$.
- The slope of edge $(c_{\beta_{s-1}}, c_{\beta_{s-1}+1})$ is in the range $(0, +1)$.
- All other slopes on C_{k-1} are not changed.
- The slopes of the incident edges of V_k are in the range $(-\infty, -1] \cup [+1, \infty)$.

This implies that the faces F_1, \dots, F_{s-1} are convex when inserting V_k at step k . To preserve convexity during the other steps $k' > k$, we add edges from c_{i_j} to $c_{\beta_j}, \dots, c_{i_{j-2}}$ ($1 \leq j \leq s$). This does not destroy planarity and implies that if c_{i_j} is shifted to right in some step $k' > k$, then also $c_{\beta_j+1}, \dots, c_{i_{j-1}}$ is shifted to right with the same value. The modified graph is still called G . Now we can prove the following lemma.

Lemma 4.2 *The faces remain convex during the algorithm.*

Proof: Assume $V_k = \{z_1\}$, and let c_{i_j} ($1 \leq j \leq s$), $c_{\alpha_j}, c_{\beta_j}$ and F_{i_j} ($1 \leq j < s$) be as defined above. (The proof is analogous when $|V_k| > 1$.) Consider a step $k' > k$. If c_{i_s} is shifted to the right, then by the added dummy edges, also $c_{\beta_{s-1}+1}, c_{i_{s-1}}$ are shifted to the right with the same value, thereby preserving planarity in the only relevant face, F_{s-1} . If z_1 is shifted to right, then the vertices c_{i_2}, \dots, c_{i_s} are shifted to right as well, and if c_{i_j} is shifted, then $c_{\beta_{j-1}+1}, \dots, c_{i_{j-1}}$ and $c_{i_{j+1}}, c_{\beta_{j+1}}$ are shifted to right with the same value. This yields planarity in the faces F_1, \dots, F_{s-1} . It also has the consequence that if c_i is shifted to right ($i_1 \leq i < i_s$), then $c_{i'}$, $i < i' \leq i_s$ is shifted to right with at least the same value.

We use this observation for the case that c_{i_1} is shifted to right in some later step k . Then z_1 is also shifted to right by at least the same value. Hence the vertices

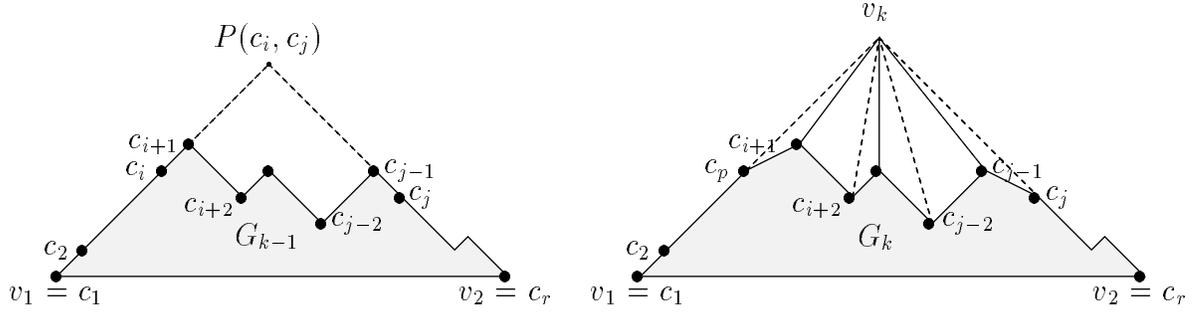


Figure 3: Idea of the straight-line drawing algorithm.

```

for  $k := 3$  to  $n$  do
  update  $x(c_i)$  and  $shift(c_r)$ ;
   $shift(c_r) := shift(c_j) + 2$ ;
   $P(v_k) := \mu((x(c_i), y(c_i)), (x(c_r) + shift(c_j), y(c_j)))$ 
rof;
 $shift(v) := rshift(v) := 0$  for all  $v \in V$ ;
for  $k := n$  downto  $2$  do
  for every internal vertex  $v_i$  of  $v_k$  do  $shift(v_i) := shift(v_k) + rshift(v_k) + 1$  rof;
   $rshift(c_r) := rshift(c_j) + rshift(v_i) + 2$ ;
   $x(v_k) := x_{insert}(v_k) + shift(v_k) + rshift(v_k)$ 
rof;
END LINEAR STRAIGHT-LINE DRAW

```

Moreover, using the *lmc*-ordering it is already sufficient that the planar graph is triconnected, because when adding $V_k = \{z_1, \dots, z_\ell\}$, we can draw z_1, \dots, z_ℓ on a horizontal line with distance two in between. This yields that edge (c_l, z_1) has a slope $+1$, edges $(z_1, z_2), \dots, (z_{\ell-1}, z_\ell)$ have slope 0 and length 2 , and edge (z_ℓ, c_r) has a slope -1 . It is easy to see that this still gives a correct straight-line drawing on an $(2n - 4) \times (n - 2)$ grid.

In the remaining part of this section we modify this new algorithm a little such that all interior faces are convex.

Let V_k be $\{z_1\}$, and let c_{i_1}, \dots, c_{i_s} be the vertices on C_{k-1} , adjacent to z_1 . ($i_1 = l$ and $i_s = r$.) Let F_j ($1 \leq j < s$) be the face formed by the edges $(z_1, c_{i_j}), (z_1, c_{i_{j+1}})$ and the path B_j between c_{i_j} and $c_{i_{j+1}}$.

Lemma 4.1 *Each path B_j has the following pattern:*

- From c_{i_j} to some vertex c_{α_j} , a sequence D_j , $|D_j| \geq 1$, of vertices with strictly decreasing y -coordinate.
- Two vertices $c_{\alpha_j}, c_{\beta_j}$ with same y -coordinate.
- From c_{β_j} to $c_{i_{j+1}}$, a sequence U_j of vertices with strictly increasing y -coordinate.

(In [7] another linear implementation of [17] is described, assuming that the input graph is triangulated.) Moreover, we will show that this algorithm can be modified such that we can draw every triconnected planar graph with convex faces on a grid.

The algorithm of [17] is as follows: it maintains a straight-line embedding during every step k of the *lmc*-ordering such that

1. v_1 is at $(0, 0)$, v_2 is at $(2k - 4, 0)$.
2. If $v_1 = c_1, c_2, \dots, c_r = v_2$ is the outerface of G_k in step k , then $x(c_1) < x(c_2) < \dots < x(c_r)$.
3. The edges (v_l, v_{l+1}) have slopes $+1$ or -1 .

Assume first that G is triangulated, in which case we can add a vertex v_k in every step k of the *lmc*-ordering [17]. Let $L(v)$ be a set of vertices. The idea of the algorithm is the following: when we add vertex v_k with leftvertex c_l and rightvertex c_r then all vertices c_{l+1}, \dots, c_{r-1} are shifted one to the right, and the vertices c_r, \dots, c_r are shifted two to the right (and of course, several internal vertices of G_{k-1} have to be shifted to the right as well). The crossing point of the line with slope $+1$ from c_l and the line with slope -1 from c_r denotes the place for vertex v_k . All vertices c_l, \dots, c_r are visible from this point, see Figure 3 for the corresponding picture. In particular, the algorithm is as follows:

{ In every step k , let c_1, \dots, c_r be the outerface,
and c_l and c_r are the left- and rightvertex of v_k , resp.}
let $\mu(p_1, p_2)$ be the crossing point of line of slope $+1$ from p_1
and line of slope -1 from p_2 .

```

P(v1) := (0, 0); L(v1) := {v1};
P(v2) := (2, 0); L(v2) := {v2};
P(v3) := (1, 1); L(v3) := {v3};
for  $k := 4$  to  $n$  do
  for  $v \in \bigcup_{l=j}^r L(c_l)$  do  $x(v) := x(v) + 2$  rof;
  for  $v \in \bigcup_{l=i+1}^{j-1} L(c_l)$  do  $x(v) := x(v) + 1$  rof;
   $P(v_k) := \mu((x(c_l), y(c_l)), (x(c_r), y(c_r)))$ ;
   $L(v_k) := \{v_k\} \cup \bigcup_{l=i+1}^{j-1} L(c_l)$ 
rof

```

The correctness of this algorithm is proved in [17]. Shiftvalues of 1 and 2 occur in the algorithm and we update the corresponding variables *shift*(v) and *rshift*(v) in a similar way. The complete algorithm can now be implemented as follows:

```

LINEARSTRAIGHT-LINEDRAW( $G$ );
P(v1) := P(v2) := (0, 0);

```

Proof: When we insert V_k in step k with leftvertex c_l and rightvertex c_r , extra time is required for walking towards c_1 to find the first *true* marked $correct(c_\alpha)$. All *correct*-values of the vertices c_α, \dots, c_{r-1} are marked *true* after visiting them. If in a step $k' > k$, $correct(c_\beta)$ becomes *false* again (with $\alpha \leq \beta \leq r - 1$), then a vertex set $V_{k'}$ with rightvertex $c_{k'} = c_\beta$ is added, with $k' \leq l$. This contradicts the *lmc*-ordering, hence every $correct(c_\beta)$ becomes once *false* and becomes *true* after visiting c_β again. Updating requires constant time, hence the total time for visiting the *false* marked vertices and updating $shift(v), x(v)$ and $correct(v)$ for all vertices is $\mathcal{O}(n)$. \square

These three lemmas show that in any step k we can compute the up-to-date x -coordinates of the vertices c_1, \dots, c_r of C_{k-1} when adding V_k , where c_r is the rightvertex of V_k . Let $P(v) = (x_{insert}(v), y_{insert}(v))$ be the coordinates of v at the time of adding v .

However, how can we compute the final x -coordinates of the vertices. Indeed, hereto we have to traverse the vertices of V_k in decreasing order, i.e., from V_k to V_1 , and set initially $shift(v) = 0$ for all $v \in V$. When considering the vertices of $V_k = \{z_1, \dots, z_\ell\}$, we set $shift(c_i) = shift(z_1)$, with $l < i < r$, and c_l and c_r the left- and rightvertex, resp., of V_k , and c_1, \dots, c_q the outerface C_{k-1} of G_{k-1} . $shift(c_l)$ is not updated (because c_l is not shifted when adding V_k initially). Since c_r is also part of some outerface $C_{k'-1}$, $k' > k$, $shift(c_r)$ could already be greater than zero at the moment of visiting V_k . The question arises whether this value was also added to the vertices of V_k or not. If this was the case, then this shift-value should not be added to $shift(c_r)$ again. How can we solve this problem?

The solution is as follows: to compute the right shift of c_r we distinguish the shifts added to c_{l+1}, \dots, c_{r-1} and to c_r , by introducing a new variable, $rshift(v)$. When considering V_k for computing the final x -coordinates, and c_r must be shifted a value x' more to the right than c_{r-1} , then we add $rshift(z_1) + x'$ to $rshift(c_r)$. The final coordinates of the vertices z_1, \dots, z_ℓ of V_k is now given by $x_{insert}(z_i) + shift(z_i) + rshift(z_i), 1 \leq i \leq \ell$.

The technique for computing the final coordinates corresponds to the idea of computing the insert-coordinates: when the vertices of V_k are shifted to the right in a later step, then also the vertices c_{l+1}, \dots, c_r must be shifted to the right as well.

The precise values for $shift$ and $rshift$ will be given in the next sections, when we present the applications for several graph drawing representations. We call this method the *shift*-method.

4 Convex drawings

The *lmc*-ordering is a generalization of the canonical ordering of de Fraysseix et al. [17]. We can apply the *lmc*-ordering and the shift-method to get a linear implementation of the straight-line grid drawing algorithm of triangulated planar graphs [17].

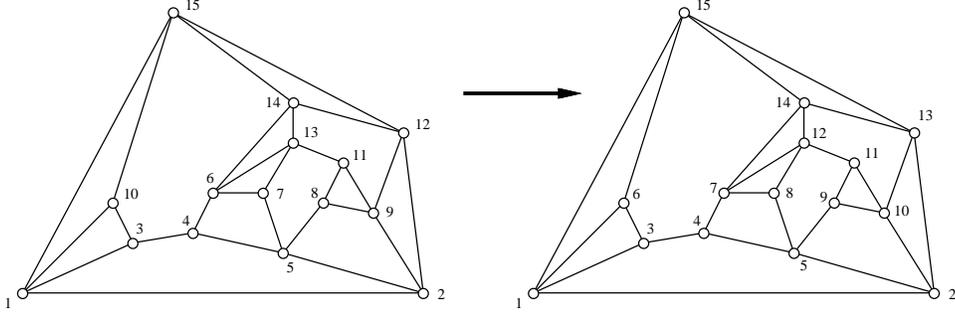


Figure 2: From a canonical ordering of the graph in Figure 1 to an *lmc*-ordering.

has been recalculated. We also introduce a counter for each vertex v , called $shift(v)$. $shift(c_r)$ denotes the value, which must be added to all $x(c_k)$, with $j \leq k \leq r$, where c_1, \dots, c_r is the current outerface C_k . When a vertex v is added by the *lmc*-ordering, we set $correct(v) = false$ and $shift(v) = 0$. Inspect step k . Let c_r be the rightvertex of V_k . We walk along the outerface from c_r towards c_1 until we find the first *true* marked $correct(c_\alpha)$. Then we walk back from c_α to c_r . When visiting c_β ($\alpha < \beta < r$) we add $\sum_{\alpha < i \leq \beta} shift(c_i)$ to $x(c_\beta)$ and set $correct(c_\beta)$ to *true*, because $x(c_\beta)$ is recalculated. We add $\sum_{\alpha < i < r} shift(c_i)$ to $shift(c_r)$. This approach is correct since the following two lemmas hold in step k :

Lemma 3.4 *All vertices c_β , $\alpha < \beta \leq r$, have $correct(c_\beta) = false$.*

Proof: Suppose not. Inspect the first time that a vertex c_γ on the outerface is encountered for which $correct(c_\gamma) = false$ and $correct(c_{\gamma+1}) = true$. $correct(c_{\gamma+1}) = true$ means that in a step $k' < k$, $x(c_{\gamma+1})$ and $correct(c_{\gamma+1})$ are recomputed, due to the insertion of a vertex set $V_{k'}$ with leftvertex $c_{l'}$, and $l' \geq \gamma + 1$. But $correct(c_\gamma) = false$ means that in step k we add V_k with rightvertex c_γ . This contradicts the definition of the *lmc*-ordering. \square

Lemma 3.5 *$r > \alpha$ holds for rightvertex c_r .*

Proof: Suppose not. $correct(c_\alpha) = true$ means that in a step $k' < k$, we updated $x(c_\alpha)$, due to the insertion of $V_{k'}$ with leftvertex $c_{l'}$, $l' \geq \alpha$. Adding V_k with rightvertex c_r in step k implies $l' > r$. Since $k' < k$, this contradicts the definition of the *lmc*-ordering. \square

Lemma 3.6 *The total time for visiting the false marked vertices and updating $shift(v)$, $x(v)$ and $correct(v)$ for all vertices v is $\mathcal{O}(n)$.*

all x -coordinates of the vertices in G_k in each step implies a quadratic running time. To avoid this, we use *lazy evaluation*:

We compute the exact coordinates of a vertex only when they are necessary to compute the coordinates of other vertices. This means that only the exact coordinates of the vertices on the outerface are essential during the insertions. As a first step towards this process, we refine the canonical ordering to the *leftmost* canonical ordering, which we will call the *lmc*-ordering from now on.

Definition 3.1 *A canonical ordering is a leftmost canonical (lmc-)ordering if we can add in any step k a vertex set V_k with leftvertex c_l or a vertex set $V_{k'}$ with leftvertex $c_{l'}$, and $l < l'$ holds, then $k < k'$.*

In other words, we take this vertex set V_k , for which the corresponding leftvertex c_l is minimal with respect to l . By planarity it follows that also the corresponding rightvertex, say c_r , is minimal with respect to r .

To compute the *lmc*-ordering, we maintain a list *Outerface-Stack* for the vertices on the outerface from left to right, implemented as a stack, and initialized as $\{v_2\}$. Also the vertex sets V_k of the canonical ordering, with pointers to its left- and right-vertex are stored. Notice that V_k can be added in step k' , if all incoming edges of V_k are part of $C_{k'-1}$. We now delete vertices from the top from *Outerface-Stack*, until we find a vertex c_r on top, which is the rightvertex of a vertex set V_k , not added yet. Let $V_k = \{z_1, \dots, z_\ell\}$ from left to right, then we add z_ℓ, \dots, z_1 in this order to *Outerface-Stack*. We repeat this step with the updated *Outerface-Stack*, until all sets V_k are added. When a vertex c_r is deleted from *Outerface-Stack*, then c_r is not the rightvertex of some set V_k (which is not added yet), because otherwise all other incoming edges of V_k are left from c_i on the current outerface, which would imply that V_k could be added. Hence every vertex set V_k will be added once to the ordering. This implies that every vertex will once be added and once be deleted from *Outerface-Stack*. Since the vertices are added from left to right to *Outerface-Stack* the vertex sets V_k are added in a leftmost order.

Theorem 3.3 *Given a canonical ordering, an lmc-ordering can be computed in linear time.*

Proof: The correctness is shown above. Regarding the time complexity, every vertex v is once added to, and once deleted from *Outerface-Stack*. Testing whether vertex c_r on top of *Outerface-Stack* is the rightmost vertex of some set V_k (not added yet) requires constant time, which completes the proof. \square

In Figure 2 an example of the *lmc*-ordering is given which will serve as an example for almost all drawing algorithms, presented in this paper.

In the drawing algorithms we distinguish the insertcoordinates of v_k (when we insert v_k by the *lmc*-ordering) and the endcoordinates of v_k (in the complete drawing). We introduce a boolean variable *correct*(v), denoting whether $x(v)$, with $v \in C_k$,

For every vertex v which becomes part of F_{out} we have to compute $sepf(v)$. Consider for this problem a face F where v belongs to. We claim that F is a separation face if and only if $outv(F) \geq 3$ or if $outv(F) = 2$ and $oute(F) = 0$. This follows because precisely in these cases, F has at least two non-adjacent vertices on the outerface which makes this face a separating face. To compute $sepf(v)$ we count the number of incident faces F of v with $outv(F) \geq 3$ or $outv(F) = 2$ and $oute(F) = 0$. A face becomes at most once a separation face, because when $outv(F)$ or $oute(F)$ decreases then a vertex or edge from F is deleted and F is added to F_{out} . Every face F with $outv(F) = oute(F) + 1$ and $oute(F) \geq 2$ can be the next face in our ordering, because in this case the vertices of F , belonging to the outerface, form a consecutive sequence. Otherwise a vertex v , $v \neq v_1, v_2$, with $sepf(v) = 0$ and $visited(v) \geq 1$ (with $visited(v)$ the number of deleted neighbors of v) can be the next vertex v_k in our ordering. By Theorem 3.1, such a face or vertex exists.

The time complexity of the algorithm is the following: every vertex v has $deg(v)$ neighbors and belongs to $deg(v)$ faces. When v becomes part of F_{out} then updating $outv(F_v)$ for all incident faces of v requires $\mathcal{O}(deg(v))$ time in total. Computing $sepf(v)$ requires $\mathcal{O}(deg(v))$ time if v becomes part of F_{out} , and $\mathcal{O}(1)$ if v was already part of F_{out} and is incident to a vertex, deleted in this step. Updating $oute(F_e)$ for an edge e , which becomes part of F_{out} , requires $\mathcal{O}(1)$ time. When F becomes a separation face then $sepf$ of the other vertices of F , part of the outerface, must be increased by one. (This are at most two vertices and this happens only once, hence requires constant time.) Deleting a vertex or face can be done in time, constant in the number of deleted edges. Since $\sum deg(v) = 2m$, and $m = \mathcal{O}(n)$, this yields a linear time and space algorithm. \square

See Figure 1 for a graph with corresponding values of the variables $visited(v)$, $sepf(v)$, $outv(F)$ and $oute(F)$.

3.2 The Drawing Framework

The general idea for drawing the graph is to start with edge (v_1, v_2) , and add in step k the vertices of V_k . In step K vertex v_n is added. Assume w.l.o.g. that in step 1 v_1 is drawn most left and v_2 most right. Let $C_{k-1} : c_1 = v_1, c_2, \dots, c_q = v_2$ be the vertices from left to right on the outerface of G_{k-1} . When adding the vertices of V_k let c_l and c_r be two neighbors of V_k on G_{k-1} , with l and r as small and as big as possible, respectively. We call c_l the *leftvertex* and c_r the *rightvertex*. Edges to lower (higher) numbered neighbors of vertex v are called *incoming (outgoing) edges* of v , and $in(v)$ and $out(v)$ denote the corresponding number.

We place the vertices in such a way on the grid such that when adding V_k , the corresponding incoming edges have downwards direction. Moreover, we want to maintain the invariant that the vertices c_1, \dots, c_q of C_k remain “visible from the top” during each step. This implies that after adding V_k , vertices c_r, \dots, c_q must be “shifted to the right”, as well as several interior vertices of G_{k-1} . However, updating

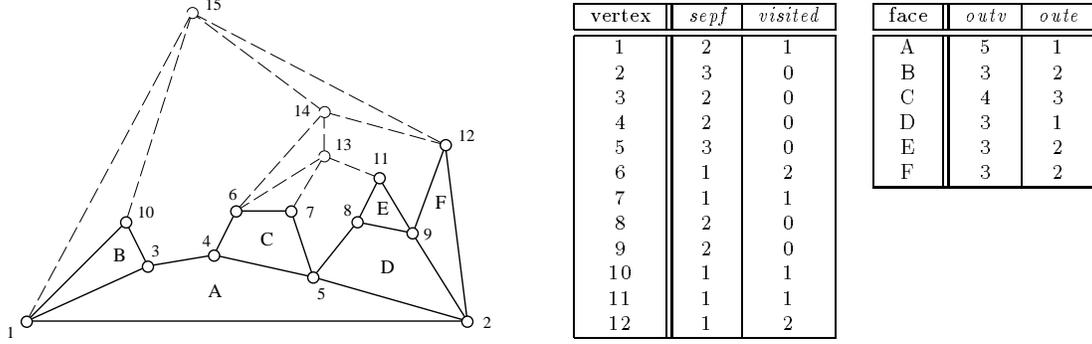


Figure 1: A graph with canonical ordering and corresponding variable-values at some step.

Assume finally that $(c_{a'}, c_{b'}) \notin G$. Let F be the face in G_k , containing the vertices $c_{a'}, c_{a'+1}, \dots, c_{b'}$. We claim that the path P in F between $c_{a'}$ and $c_{b'}$ not containing $c_{a'+1}, \dots, c_{b'+1}$, does not contain another vertex on the outerface of G_k . Suppose not, i.e, the path P contains a vertex c_d on the outerface of G_k . Suppose w.l.o.g. $1 \leq d < a'$. But now it follows that $c_{d+1}, \dots, c_{a'-1}$ is a chain of vertices of degree 2. Since $(c_d, c_{a'}) \notin G$ it follows that $\deg(c_{a'}) = 2$, which is a contradiction. Hence $G_k - \{c_{a'+1}, \dots, c_{b'+1}\}$ is biconnected and we set $V_k = \{c_{a'+1}, \dots, c_{b'+1}\}$. \square

If $V_k\{z_1, \dots, z_\ell\}$ with $\ell > 1$, then precisely one face is added, otherwise one vertex is added to G_{k-1} . The algorithm for computing the canonical ordering is based on the proof of Theorem 3.1: We start with the entire graph G , and in each step we delete a face or vertex. For this we introduce a variable $outv(F)$ and $oute(F)$ for each face F , denoting the number of vertices and edges of F belonging to the current outerface. We also introduce a variable $sepf(v)$ for every vertex v , denoting the number of different faces, containing a separation pair with vertex v . By the proof of Theorem 3.1, both v and w are part of the current outerface. We call the corresponding faces *separation faces*. Using these variables we can prove the following theorem.

Theorem 3.2 *For every triconnected planar graph a canonical ordering can be computed in linear time and space.*

Proof: Let an embedding of the triconnected planar graph G be given. Let every vertex v and edge e have pointers to the faces they belong to. Initially all variables $outv(F)$, $oute(F)$ and $sepf(v)$ are set to 0. We take an arbitrary face F_{out} as outerface during the algorithm. Assign v_1 with neighbors v_2 and v_n on F_{out} . In every step we remove vertices from G and update F_{out} . For every vertex $v \in F_{out}$ and every edge $e \in F_{out}, e \neq (v_1, v_2)$, we increase $outv(F_v)$ and $oute(F_e)$. for every $F_v \neq F_{out}$ where v belongs to, and $F_e \neq F_{out}$, where e belongs to.

- (a) V_k is a singleton, $\{z\}$, where z belongs to C_k and has at least one neighbor in $G - G_k$.
- (b) V_k is a chain, (z_1, \dots, z_ℓ) , where each z_i has at least one neighbor in $G - G_k$, and where z_1 and z_ℓ each have one neighbor on C_{k-1} , and these are the only two neighbors of V_k in G_{k-1} .

Theorem 3.1 *Every triconnected planar graph G with pre-defined v_1, v_2, v_n has a canonical ordering.*

Proof: Let G be a triconnected planar graph with v_1, v_2 and v_n given in advance. The decomposition of the vertices in V_1, \dots, V_K will be defined by reverse induction. Assume that v_2 and v_n are neighbors of v_1 and belong to the outerface of a planar embedding of G . Notice that by triconnectivity of G , the graph $G_{n-1} = G - \{v_n\}$ is biconnected and the outerface C_{n-1} is a cycle, containing (v_1, v_2) .

Let $2 < k < K$ be fixed. Assume that V_i has already been determined for every $i > k$ such that the subgraph G_k satisfies the conditions of the canonical ordering. Notice that if there are vertices $v \in G_k$ of degree 2 then $v \in C_k$. Notice also that by triconnectivity of G there are at least 3 vertices $c_\alpha, c_\beta, c_\gamma \in C_k$ having edges to vertices in $G - G_k$. Assume w.l.o.g. that $c_\alpha \neq v_1, v_2$. If G_k is triconnected then we can take $V_k = \{c_\alpha\}$ because by triconnectivity, c_α has at least three neighbors in G_k and $G_k - c_\alpha$ is biconnected.

Assume further that G_k is not triconnected, hence G_k contains separation pairs. Let v_x, v_y be a separation pair, and let G_1, G_2 be two components of $G_k - \{v_x, v_y\}$. Since G is triconnected, there is a path P between G_1 and G_2 , not visiting v_x and v_y . In G_k , v_x and v_y are forming a separation pair, hence the edges of path P are removed in G_k . Since we defined the ordering by reverse induction, we removed only vertices and edges from the outerface. Hence path P goes between two vertices $c_{x'}, c_{y'}$, belonging to C_k with $c_{x'} \in G_1$ and $c_{y'} \in G_2$. This yields that v_x and v_y belong to C_k and one path between v_x and v_y on C_k is part of G_1 ; the other path on C_k between v_x and v_y is part of G_2 . This holds for every separation pair v_x, v_y , hence all vertices of the separation pairs belong to C_k .

Let c_a, c_b be a separation pair such that $b - a$ is minimal. If $\deg(c_{a+1}) > 2$ then there is a vertex c_α , $a < \alpha < b$, with at least one edge to a vertex deleted in step $j > k$, otherwise the graph $G - \{c_a, c_b\}$ is disconnected, which contradicts the triconnectivity of G . By minimality of $b - a$, c_α is not part of a separation pair in G_k , hence $G_k - c_\alpha$ does not have a cutvertex and the outerface of $G_k - c_\alpha$ is biconnected. We take $V_k = \{c_\alpha\}$ in the ordering.

Assume now that there is no separation pair c_a, c_b with $\deg(c_{a+1}) > 2$. Then $b = a + 2$, because c_a and c_{a+2} are the only neighbors of c_{a+1} in G_k . Let now $1 \leq a' \leq a$ and $b \leq b' \leq r$ be such that all vertices $c_{a'+1}, c_{a'+2}, \dots, c_{b'-1}$ have degree 2, and $\deg(c_{a'}) > 2$ if $a' > 1$ and $\deg(c_{b'}) > 2$ if $b' < r$. Notice that $v_1, v_2 \notin \{c_{a'+1}, \dots, c_{b'-1}\}$ and every vertex $c'_i, a' < i < b'$, has an edge to $G - G_k$. If edge $(c_{a'}, c_{b'}) \in G$, then $G_k - \{c_{a'+1}, \dots, c_{b'-1}\}$ is biconnected and we set $V_k = \{c_{a'+1}, \dots, c_{b'-1}\}$.

from G preserves the connectivity. 2- and 3-connected are also called *biconnected* and *triconnected*, resp. A vertex, whose deletion disconnects the graph is called a *cutvertex*. A set of two vertices, whose deletion disconnects a graph is called a *separation pair*. It is well known that if the planar graph is triconnected, then its embedding is unique (up to choosing an exterior face). A *path* between two vertices x and y is an alternating sequence of vertices and edges such that x and y are at the end of this sequence and each edge in the sequence is preceded and followed by its end vertices. If the vertices on the path have degree two then the path is called a *chain*.

A drawing such that each edge is represented by a polygonal chain is a *polygonal* drawing. There are two common special cases of this standard. A *straight-line* drawing maps each edge into a straight-line segment. An *orthogonal* drawing maps each edge into a chain of horizontal and vertical segments. Note that polyline drawings can be modified to give drawings with nicely curved edges. A polyline drawing is a *grid* drawing if the vertices and the bends of the edges have integer coordinates (see also [9] for more definitions and figures, not presented here). In our drawing algorithms we denote the place of vertex v by $P(v) = (x(v), y(v))$. Several other definitions and explaining figures, not presented here, are presented in the following sections when we need them.

3 The Canonical Ordering and the Drawing Framework

3.1 The Canonical Ordering

In this section we introduce the *canonical ordering* for triconnected planar graphs. In the next section we refine it to a *leftmost* canonical ordering, to get a linear time drawing framework.

(Canonical Ordering)

Let G be a triconnected plane graph with an edge (v_1, v_2) on the external face. Let $\pi = (V_1, \dots, V_K)$ be an ordered partition of V , that is, $V_1 \cup \dots \cup V_K = V$ and $V_i \cap V_j = \emptyset$ for $i \neq j$. Define G_k to be the subgraph of G induced by $V_1 \cup \dots \cup V_k$, and denote by C_k the external face of G_k . We say that π is a *canonical ordering* of G if:

- V_1 consists of $\{v_1, v_2\}$.
- V_K is a singleton $\{v_n\}$, where v_n lies on the outerface and is a neighbor of v_1 .
- Each C_k ($k > 1$) is a cycle containing (v_1, v_2) .
- Each G_k is biconnected and internally triconnected, that is, removing two internal vertices of G_k does not disconnect it.
- For each k in $2, \dots, K - 1$, one of the two following conditions holds:

1. Every triconnected planar graph can be drawn convexly with straight lines on a grid of size at most $(2n - 4) \times (n - 2)$. This is the first algorithm combining the aspects of grid size with convexity. It also implies a new and rather simple proof that every triconnected planar graph admits a convex drawing.
2. Every triconnected 4-planar graph can be drawn with at most $\lceil \frac{3}{2}n \rceil + 4$ bends on an $n \times n$ grid. If $n > 6$ then every edge is drawn with at most two bends.
3. Every 3-planar graph G can be drawn with at most $\lfloor \frac{n}{2} \rfloor + 1$ bends on an $\lfloor \frac{n}{2} \rfloor \times \lfloor \frac{n}{2} \rfloor$ grid. A nice characteristic is that G has a spanning tree using $n - 1$ straight-line edges and all non-tree edges have at most one bend.
4. Every triconnected planar graph can be drawn on a grid of size at most $(2n - 6) \times (3n - 9)$, with minimum angle $> \frac{2}{3}$ and at most $5n - 15$ bends totally (we call this the *mixed model*). Every edge has at most three bends. According to our opinion, this is the first practical drawing algorithm, having good bounds on the area, number of bends, and minimum angle. The result is extended to general planar graphs as well.
5. A new and simple method for computing a visibility representation of a planar graph on a grid of size at most $(2n - 5) \times (n - 1)$.

The paper is organized as follows: in section 2 we give some necessary definitions. In section 3 we present the canonical ordering and the general drawing framework. In section 4, 5, 6 and 7 we present the drawing results for convex drawing, orthogonal drawing, the mixed model and for the visibility representation, respectively. Section 8 contains some final remarks and open questions.

2 Definitions

Let $G = (V, E)$ be a planar graph with n vertices and m edges. A graph is called *planar* if it can be drawn without any pair of crossing edges. A *planar embedding* is a representation of a planar graph in which at every vertex all edges are sorted in clockwise order when visiting them around the vertex with respect to the planar drawing. The embedding divides the plane into a number of *faces*. The unbounded face is the *exterior face* or *outerface*, all other faces are called *interior faces*. Edges and vertices, belonging to the outerface, are called *exterior edges* and *exterior vertices*, respectively; the other edges and vertices are called *interior edges* and *interior vertices*. An interior edge, connecting two exterior vertices, is called a *chord*.

$\deg(v)$ denotes the degree of v , i.e., the number of neighbors of vertex v . A *k-planar* graph is a planar graph with maximum degree k . The *induced subgraph* on the vertices v_1, \dots, v_k consists of the vertices v_1, \dots, v_k and the edges (v_i, v_j) with $1 \leq i, j \leq k$. $G - \{v\}$ denotes the graph after deleting vertex v and its incident neighbors from G . G is called *k-connected* if deleting any $k - 1$ vertices

1 Introduction

The problem of “nicely” drawing a graph in the plane is an emerging area of research that combines flavors of topological graph theory and computational geometry. The large number of applications include VLSI layout, algorithm animation, visual languages and CASE tools. Several criteria to obtain a high aesthetic quality have been established. Typically, vertices are represented by distinct points in a line or plane, and are sometimes restricted to be grid points. Alternatively, vertices are sometimes represented by line segments. Edges are often constrained to be drawn as straight lines or as a contiguous set of line segments (e.g. when bends are allowed). The objective is to find a layout for a graph that optimizes some cost function, such as area, minimum angle, number of bends, or satisfies some other constraint. See the annotated bibliography for an up to date overview of the recent developments and optimization problems in graph drawings with more than 300 references [9].

It is well-known that every planar graph can be drawn planar with straight lines, and by more recent algorithms this can be done in linear time and space on a grid of size $\mathcal{O}(n) \times \mathcal{O}(n)$ (see e.g. [7, 17, 19, 34]). $\Omega(n^2)$ is also a lower bound for the area of planar straight-line drawings [17]. However, a drawback of all these drawing algorithms is that the minimum angle between lines can be very small which makes the drawing unattractive. Therefore several papers investigate the extra criterion of *convexity*: every face must be drawn convexly. Tutte showed that every triconnected planar graph can be drawn with convex faces. Thomassen characterized the class of planar graphs which admit a convex drawing, and Chiba et al. [5] presented a linear time drawing algorithm for this class. However, the coordinates of the vertices can be reals and a huge number of vertices can be clustered in a small area.

Another representation model is the *orthogonal representation*, in which the vertices are represented by points and edges by alternately horizontal and vertical segments, connecting the endpoints. Orthogonal drawings have numerous important applications in the field of VLSI-design and graphics. Storer [35] and Tamassia & Tollis [37] presented linear time heuristics to construct an orthogonal representation of a 4-planar (3-planar) graph with at most $2n + 4$ bends ($n + 2$ bends, resp.). Every edge is bent at most four times. Tamassia presented an $\mathcal{O}(n^2 \log n)$ time algorithm for drawing an *embedded* 4-planar graph on an $n \times n$ grid with the minimum number of bends [36]. If the planar embedding is not given in advance, then the problem is polynomial time solvable for 3-planar graphs [12], and NP-hard for 4-planar graphs [18]. In particular, Garg & Tamassia showed that is even NP-hard to approximate the minimum number of bends in a planar orthogonal drawing with an $\mathcal{O}(n^{1-\epsilon})$ error, for any $\epsilon > 0$ [18].

In this paper we introduce a new ordering on the vertices and faces of a triconnected planar graph, called the *canonical ordering*. The canonical ordering can be computed in linear time. By refining the canonical ordering to a *leftmost* canonical ordering (or *lmc*-ordering), this leads to a general framework for drawing triconnected planar graphs on a grid, and implies several drawing results:

Drawing Planar Graphs Using the Canonical Ordering*

Goos Kant[†]

Dept. of Computer Science, Utrecht University
P.O. Box 80.089, 3508 TB Utrecht, the Netherlands

Abstract

We introduce a new method to optimize the required area, minimum angle and number of bends of planar drawings of graphs on a grid. The main tool is a new type of ordering on the vertices and faces of triconnected planar graphs. Using this method linear time and space algorithms can be designed for many graph drawing problems.

- Every triconnected planar graph G can be drawn convexly with straight lines on an $(2n - 4) \times (n - 2)$ grid, where n is the number of vertices.
- Every triconnected planar graph with maximum degree four can be drawn orthogonally on an $n \times n$ grid with at most $\lceil \frac{3n}{2} \rceil + 4$, and if $n > 6$ then every edge has at most two bends.
- Every 3-planar graph G can be drawn with at most $\lfloor \frac{n}{2} \rfloor + 1$ bends on an $\lfloor \frac{n}{2} \rfloor \times \lfloor \frac{n}{2} \rfloor$ grid.
- Every triconnected planar graph G can be drawn planar on an $(2n - 6) \times (3n - 9)$ grid with minimum angle larger than $\frac{2}{d}$ radians and at most $5n - 15$ bends, with d the maximum degree.
- There is a new method for constructing a visibility representation of a planar graph on a grid of size at most $(2n - 5) \times (n - 1)$.

These results give in some cases considerable improvements over previous results, and give new bounds in other cases. Several other results, e.g. concerning visibility representations, are included.

*This work was supported by the ESPRIT Basic Research Actions program of the EC under contract No. 7141 (project ALCOM II). An extended abstract of this paper was presented at the *33th Annual IEEE Symp. on Found. of Comp. Science*, Pittsburgh, 1992. This is a revised version of Technical Report RUU-CS-92-33.

[†]Email: goos@cs.ruu.nl

ISSN: 0924-3275

Drawing Planar Graphs Using the Canonical Ordering

Goos Kant

Dept. of Computer Science, Utrecht University
P.O. Box 80.089, 3508 TB Utrecht, the Netherlands

Technical Report RUU-CS-92-33
October 1992

Department of Computer Science
Utrecht University
P.O.Box 80.089
3508 TB Utrecht
The Netherlands

Drawing Planar Graphs Using the Canonical Ordering

Goos Kant

Dept. of Computer Science, Utrecht University
P.O. Box 80.089, 3508 TB Utrecht, the Netherlands

RUU-CS-92-33

October 1992



Utrecht University

Department of Computer Science

Padualaan 14, P.O. Box 80.089,
3508 TB Utrecht, The Netherlands,
Tel. : + 31 - 30 - 531454