

Algorithms for Drawing Plane Graphs

Takao NISHIZEKI[†], Member, Kazuyuki MIURA[†], and Md. Saidur RAHMAN[†], Nonmembers

SUMMARY Graph drawing addresses the problem of constructing geometric representation of information and finds applications in almost every branch of science and technology. Efficient algorithms are essential for automatic drawings of graphs, and hence a lot of research has been carried out in the last decade by many researchers over the world to develop efficient algorithms for drawing graphs. In this paper we survey the recent algorithmic results on various drawings of plane graphs: straight line drawing, convex drawing, orthogonal drawing, rectangular drawing and box-rectangular drawing.

key words: straight line drawing, convex drawing, grid drawing, rectangular drawing, box-rectangular drawing, orthogonal drawing, plane graph

1. Introduction

Recently automatic aesthetic drawing of graphs has created intense interest due to their broad applications, and as a consequence, a number of drawing methods have come out [3]–[11], [14], [16], [29], [33]. The most typical method is the *straight line drawing* in which all edges of a graph are drawn as straight line segments without any edge-intersection, as illustrated in Figs. 1 and 2(d). Every plane graph has a straight line drawing [10], [31], [34]. However, not every straight line drawing is an aesthetic drawing. Another typical method which often produces an aesthetic drawing is a *convex drawing* in which every face boundary is drawn as a convex polygon. Not every plane graph has a convex drawing, but every 3-connected plane graph has a convex drawing.

In the last few years a substantial research on graph drawing has been devoted to “orthogonal drawings” for their numerous applications in circuit schematics, entity relationship diagrams, data flow diagrams etc. [9]. An *orthogonal drawing* of a plane graph G is a drawing of G with the given embedding in which each vertex is mapped to a point, each edge is drawn as a sequence of alternate horizontal and vertical line segments, and any two edges do not cross except at their common end, as illustrated in Fig. 9(b). An orthogonal drawing is called a *rectangular drawing* if it has no bend and each face is drawn as a rectangle, as illustrated in Fig. 5(b). A rectangular drawing has practical applications in VLSI floorplanning [18], [19], [25], [28] and architectural floorplanning [20]. A generalization of a rectangular drawing called a *box-rectangular drawing* has been emerged to meet some floorplanning requirements, where each vertex is drawn as a rectangular box [27]. (See Fig. 8(e).)

In this paper we survey the recent results on the straight line drawings, convex drawings, rectangular drawings, box-rectangular drawings and orthogonal drawings of plane graphs.

2. Straight Line Grid Drawing

A straight line drawing of a plane graph G is called a *grid drawing* of G if the vertices of G are put on grid points of integer coordinates. Of course, the distance between any two vertices in the drawing is at least 1. The *integer grid* of size $W \times H$ consists of $W + 1$ vertical segments and $H + 1$ horizontal segments, and has a rectangular contour. W and H are called the *width* and *height* of the integer grid, respectively. It is known that every plane graph of $n \geq 3$ vertices has a grid drawing on an $(n - 2) \times (n - 2)$ grid, and that such a grid drawing can be found in linear time [5], [7], [11], [29]. It is also shown that, for each $n \geq 3$, there exists a plane graph which needs a grid of size at least $\lceil 2(n - 1)/3 \rceil \times \lceil 2(n - 1)/3 \rceil$ for any grid drawing [6], [11]. It has been conjectured that every plane graph has a grid drawing on a $\lceil 2n/3 \rceil \times \lceil 2n/3 \rceil$ grid, but it is still an open problem. On the other hand, a restricted class of graphs has a more compact grid drawing. For example, if G is a 4-connected plane graph and has at least four vertices on its outer face, then G has a grid drawing on a $W \times H$ grid such that $W + H \leq n$, $W \leq (n + 3)/2$ and $H \leq 2(n - 1)/3$, and one can find such a grid drawing in linear time [14]. However, the algorithm by X. He [14] is rather complicated.

Miura *et al.* [21] recently give a very simple algorithm which finds a grid drawing of any given 4-connected plane graph G on a $W \times H$ grid such that $W = \lceil n/2 \rceil - 1$ and $H = \lceil n/2 \rceil$ in linear time if G has four or more vertices on the outer face. Since $W = \lceil n/2 \rceil - 1$ and $H = \lceil n/2 \rceil$, $W + H \leq n$. Thus their bounds on W and H are better than X. He's bounds [14]. Their bounds are indeed best possible, because there exist an infinite number of 4-connected plane graphs, for example the nested quadrangles depicted in Fig. 1, which need grids of size at least $W = \lceil n/2 \rceil - 1$ and $H = \lceil n/2 \rceil$ for any grid drawing. An aspect ratio of a drawing obtained by the algorithm in [14] may be $1 : 4/3$, while the ratio obtained by the algorithm in [21] is always $1 : 1$.

The outline of the algorithm of Miura *et al.* [21] is as follows. One can assume without loss of generality that a given graph G is internally triangulated as illustrated in Fig. 2(a). First, the algorithm finds a “4-canonical ordering” of G [17]. Using the ordering, the algorithm then di-

Manuscript received February 26, 2003.

[†]The authors are with the Graduate School of Information Sciences, Tohoku University, Sendai-shi, 980-8579 Japan.

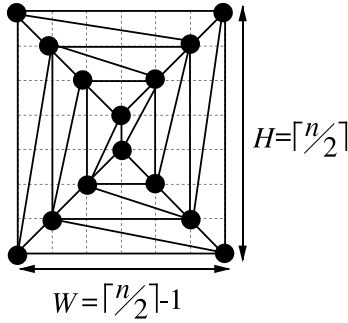


Fig. 1 Nested quadrangles attaining the bounds.

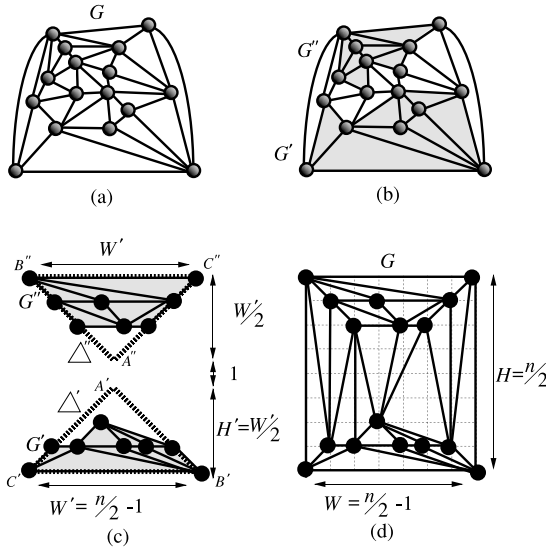


Fig. 2 Drawing process of algorithm.

vides G into two graphs G' and G'' , each of which has about $n/2$ vertices as illustrated in Fig. 2(b) where G' and G'' are shaded. Next, the algorithm draws the plane subgraph G' in an isosceles right-angled triangle whose base has length $W' = n/2 - 1$ and whose height is $H' = W'/2$, as illustrated in Fig. 2(c). Similarly, the algorithm draws G'' in the same triangle with its upside down. In Fig. 2(c) the two triangles are drawn by thick dotted lines. The algorithm places the two triangles so that their vertices opposite to their bases are separated by distance 1. Finally, the algorithm combines the drawings of G' and G'' to obtain a grid drawing of G , as illustrated in Fig. 2(d). The drawing of G has sizes $W = W' = n/2 - 1$ and $H = 2H' + 1 = W' + 1 = n/2$.

3. Convex Grid Drawing

A straight line drawing of a plane graph G is often aesthetically pretty if every face boundary is drawn as a convex polygon [3], [33]. Such a drawing is called a *convex drawing* of G . Not every plane graph has a convex drawing, but every 3-connected plane graph has a convex drawing [33]. Thomassen obtained a necessary and sufficient condition for a plane graph to have a convex drawing [32]. Chiba *et al.*

gave a linear algorithm to examine the condition and find a convex drawing [3], [4]. They also gave a linear algorithm to obtain a plane embedding of a given *planar* graph which has a convex drawing [4]. A convex drawing is called a *convex grid drawing* if it is a grid drawing. Every 3-connected plane graph has a convex grid drawing on an $(n-2) \times (n-2)$ grid, and that such a grid drawing can be found in linear time [5], [30]. The size of an integer grid required by a convex grid drawing would be smaller than $(n-2) \times (n-2)$ for 4-connected plane graphs, but it has not been known how small the grid size is.

Miura *et al.* [22] recently give an answer to this problem. That is, they give an algorithm which finds in linear time a convex grid drawing of any given 4-connected plane graph $G = (V, E)$ on an integer grid such that $W + H \leq n - 1$ if G has four or more vertices on the outer face boundary. Since $W + H \leq n - 1$, $W \times H \leq \lceil (n-1)/2 \rceil \cdot \lfloor (n-1)/2 \rfloor$. The outer face boundary of G is always drawn as a rectangle as illustrated in Fig. 4. Their bounds on $W + H$ and $W \times H$ are optimal in a sense that there exist an infinite number of 4-connected plane graphs, for example the nested quadrangles depicted in Fig. 1, which need grids such that $W + H = n - 1$ and $W \times H = \lceil (n-1)/2 \rceil \cdot \lfloor (n-1)/2 \rfloor$. Thus the area of an integer grid can be reduced to 1/4 and the contour length to half for 4-connected plane graphs than those for 3-connected plane graphs. It should be noted that any 4-connected plane graph G with four or more vertices on the outer face boundary has a grid drawing on an almost square grid with $W = \lceil n/2 \rceil - 1$ and $H = \lceil n/2 \rceil$ [21], but the drawing is not always convex.

In the reminder of the section we present the algorithm of Miura *et al.* [22]. We call the boundary of the outer face of a plane graph G the *contour* of G , and denote it by $C_o(G)$. We denote by $d(u, G)$ the degree of a vertex u in a graph G . The “4-canonical decomposition” of a plane graph G [23] playing a crucial role in the algorithm is a generalization of two well-known concepts: the “canonical ordering,” which is used to find a convex grid drawing of a 3-connected plane graph [16]; and the “4-canonical ordering,” which is used to find a “visibility representation” and a grid drawing of a 4-connected plane graph [14], [17], [21]. A 4-canonical decomposition $\Pi = (U_1, U_2, \dots, U_{12})$ is illustrated in Fig. 3(a) for a 4-connected plane graph. Let m be a natural number, and let $\Pi = (U_1, U_2, \dots, U_m)$ be a partition of set V to m subsets $U_1, U_2, \dots, U_m$ of V . Of course, $U_1 \cup U_2 \cup \dots \cup U_m = V$, and $U_i \cap U_j = \emptyset$ for any i and j , $i \neq j$. Let G_k , $1 \leq k \leq m$, be the plane subgraph of G induced by the vertices in $U_1 \cup U_2 \cup \dots \cup U_k$, and let $\overline{G}_k$ be the plane subgraph of G induced by the vertices in $U_{k+1} \cup U_{k+2} \cup \dots \cup U_m$. Thus $G = G_m = \overline{G}_0$. We say that Π is a *4-canonical decomposition* of G if the following three conditions are satisfied:

- (co1) U_1 consists of the two ends of an edge on $C_o(G)$, and U_m consists of the two ends of another edge on $C_o(G)$;
- (co2) for each k , $2 \leq k \leq m - 1$, both G_k and $\overline{G}_{k-1}$ are biconnected and

(co3) for each k , $2 \leq k \leq m-1$, one of the following three conditions holds:

- (a) U_k is a singleton set of a vertex u on $C_o(G_k)$ such that $d(u, G_k) \geq 2$ and $d(u, \overline{G_{k-1}}) \geq 2$.
- (b) U_k is a set of two or more consecutive vertices on $C_o(G_k)$ such that $d(u, G_k) = 2$ and $d(u, \overline{G_{k-1}}) \geq 3$ for each vertex $u \in U_k$.
- (c) U_k is a set of two or more consecutive vertices on $C_o(G_k)$ such that $d(u, G_k) \geq 3$ and $d(u, \overline{G_{k-1}}) = 2$ for each vertex $u \in U_k$.

We number all vertices of G by $1, 2, \dots, n$ so that they appear in $U_1, U_2, \dots, U_m$ in this order, and call each vertex in G by the number i , $1 \leq i \leq n$. Thus one can define an order $<$ among the vertices in G .

We define some terms which are used in the algorithm. We denote by $y(u)$ the y -coordinate of a vertex u , and by $x(u)$ the x -coordinate. The *lower neighbors* of u are the neighbors of u which are smaller than u . The *upper neighbors* of u are the neighbors of u which are larger than u . Every upper neighbor v of any vertex u satisfies $y(v) \geq y(u)$ in the drawing. The number of lower neighbors of u is denoted by $d_{low}(u, G)$, and the number of upper neighbors of u is denoted by $d_{up}(u, G)$. Every vertex u except vertex 1 satisfies $d_{low}(u, G) \geq 1$, and every vertex u except vertex n satisfies $d_{up}(u, G) \geq 1$. We denote by $w^*(u)$ the largest neighbor of a vertex u , $1 \leq u \leq n-1$. The *in-degree* of a vertex u in a directed graph D is denoted by $d_{in}(u, D)$, while the *out-degree* of u is denoted by $d_{out}(u, D)$.

We are now ready to present the algorithm in [22] which finds a convex grid drawing of any given 4-connected plane graph G with four or more vertices on the contour $C_o(G)$. The algorithm decides only the integer coordinates of the vertices $1, 2, \dots, n$ of G effectively in this order. One can immediately find a (straight line) grid drawing of G from the coordinates. We first show how to decide the x -coordinates of all vertices, and then show how to decide the y -coordinates.

We first show how to decide the x -coordinates of all vertices. The algorithm of Miura *et al.* puts vertices on the same vertical grid line as many as possible to reduce the width W of the drawing. Suppose that vertex i has been put on a grid point. If possible, the algorithm puts an upper neighbor j of i on the same vertical grid line as i , that is, it decides $x(j) = x(i)$ and hence $y(j) > y(i)$ of course. The algorithm tries to choose as j the largest neighbor $w^*(i)$ of i (this is crucial for making every face boundary a convex polygon). However, it is impossible for a case where $w^*(i)$ has been already put on the same vertical grid line as a vertex i' ($< i$), which was put on a grid point before i , that is, $w^*(i') = w^*(i)$. Thus, if there exist upper neighbors of i which have not been put on the same vertical grid line as any vertex i' ($< i$), then the algorithm puts the largest one j among them on the same vertical grid line as i . If there does not exist such an upper neighbor of i , then the algorithm does not put any vertex ($> i$) on the same vertical grid line as i . In this way, the following **procedure** Construct- F

constructs a directed forest $F = (V, E_F)$. All vertices in each component of F have the same x -coordinate; if there is a directed edge (i, j) in F , then $x(j) = x(i)$ and $y(j) > y(i)$.

Procedure Construct- F

begin $\{F = (V, E_F)\}$

- 1 $E_F := \emptyset$; {the initial forest $F = (V, \emptyset)$ consists of isolated vertices}

- 2 **for** $i := 1$ **to** n **do**

if vertex i has upper neighbors j such that $d_{in}(j, F) = 0$ **then**

- 3 let j be the largest one among them, and add a directed edge (i, j) to the directed graph F , that is, $E_F := E_F \cup \{(i, j)\}$;

end.

Since $d_{in}(i, F), d_{out}(i, F) \leq 1$ for each vertex i , $1 \leq i \leq n$, F is a forest and each component of F is a directed path. Clearly $d_{in}(1, F) = d_{in}(2, F) = 0$ and $d_{out}(n-1, F) = d_{out}(n, F) = 0$. Figure 3(b) illustrates the directed forest F of the graph G in Fig. 3(a). Both the path 1, 13, 20 going clockwise on $C_o(G)$ from 1 to $n-1 = 20$ and the path 2, 14, 21 going counterclockwise on $C_o(G)$ from 2 to $n = 21$ are directed paths in F , and hence these two paths are put on vertical grid lines as shown in the bottom figure of Fig. 4. Each of the other paths in F is put on a vertical grid line, too.

We then show how to arrange the paths in F from left to right. The algorithm decides a total order among all starting vertices of paths in F . For this purpose, using the following **procedure** Total-Order, the algorithm finds a directed path P going from vertex 1 to vertex 2 passing through all starting vertices of F . In Fig. 3(c), the directed path P is drawn by dotted lines.

Procedure Total-Order

begin

- 1 let P be the path directly going from vertex 1 to vertex 2;

- 2 **for** $i := 3$ **to** n **do**

if $d_{in}(i, F) = 0$ **then** $\{i$ is a starting vertex of a path in $F\}$

begin

- 3 let j be the first lower neighbor of i in the i 's adjacency list in which the i 's neighbors appear counterclockwise around i , and the first element of which is $w^*(i)$;
- 4 let j' be the starting vertex of the path in F containing vertex j ; $\{2 \neq j' < i\}$
- 5 let k be the successor of j' in path P ; {the path starting from vertex k in F has been put next to the right of the path starting from vertex j' }
- 6 insert i in P between j' and k ; {the path starting from i in F is put between the path starting from j' and the path starting from k }

end

end.

The algorithm constructs a weighted tree T rooted at

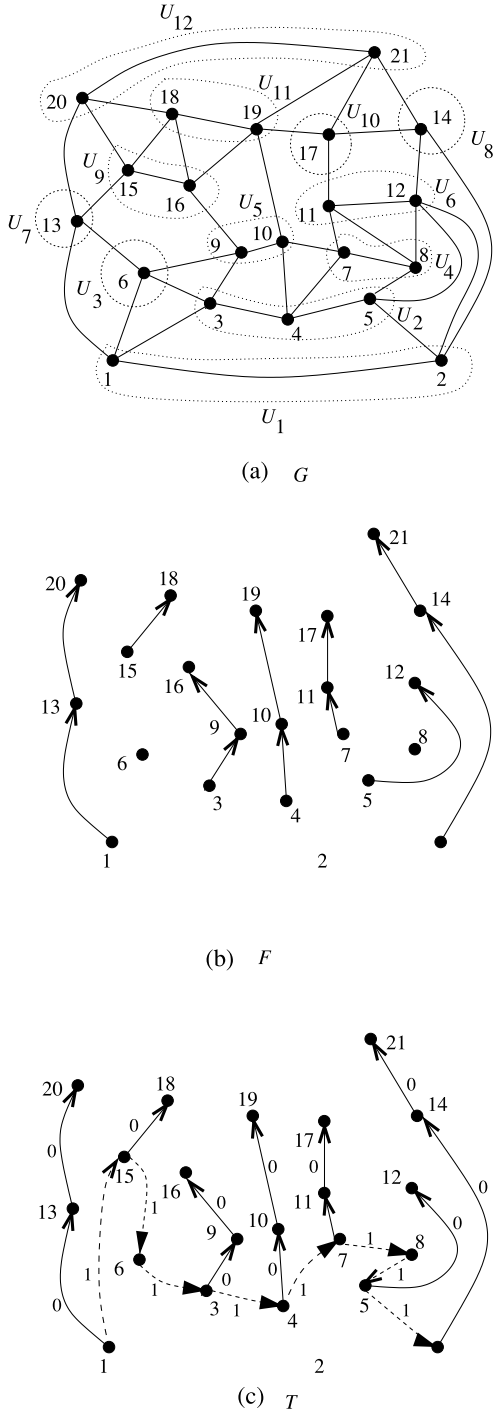


Fig. 3 Illustrations for (a) a canonical decomposition, (b) Procedure Construct- F and (c) Procedure Total-Order.

vertex 1 by adding the path P to the forest F ; every edge of F has weight 0, and every edge of P has weight 1 in T . (See Fig. 3 (c).) Then the x -coordinate $x(i)$ of each vertex i , $1 \leq i \leq n$, is the length of the path from root 1 to i in T . Thus $x(1) = 0$, and the width $W = x(2)$ of the drawing is equal to the number of paths in F except one starting from vertex 1, i.e., the number of vertices of in-degree 0 in F except vertex 1.

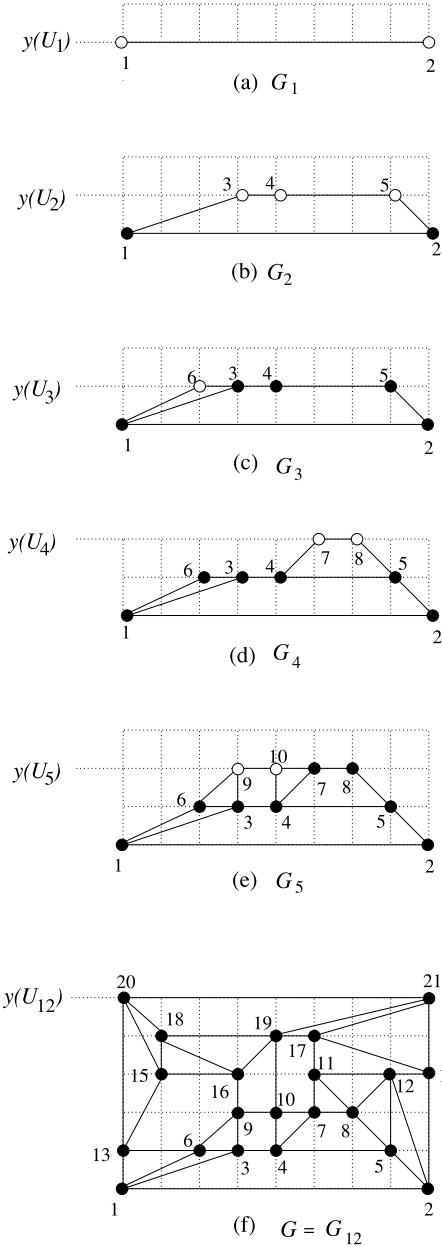


Fig. 4 Illustration of the algorithm.

We now show how to compute y -coordinates. For each index k , $1 \leq k \leq m$, y -coordinates of all vertices in $U_k = \{u_1, u_2, \dots, u_h\}$ are decided as the same integer, which is denoted by $y(U_k)$. Thus the path $u_1, u_2, \dots, u_h$ on $C_o(G_k)$ is drawn as a horizontal line segment connecting points $(x(u_1), y(U_k))$ and $(x(u_h), y(U_k))$. (See Fig. 4.) Furthermore, the algorithm decides the y -coordinates $y(U_1), y(U_2), \dots, y(U_m)$ in this order. Thus $H = y(U_m)$.

The algorithm first decides the y -coordinate $y(U_1)$ of $U_1 = \{1, 2\}$ as $y(U_1) = 0$. Thus it draws $G_1 = K_2$ as a horizontal line segment connecting points $(x(1), 0)$ and $(x(2), 0)$, as illustrated in Fig. 4 (a).

Suppose that $y(U_1), y(U_2), \dots, y(U_{k-1})$, $k \geq 2$, have al-

ready been decided, that is, G_{k-1} has already been drawn, and one is now going to decide $y(U_k)$ and obtain a drawing of G_k by adding the vertices in U_k to the drawing of G_{k-1} . Let $C_o(G_{k-1}) = w_1, w_2, \dots, w_t$, where $w_1 = 1$ and $w_t = 2$. Let $C_o(G_k) = w_1, w_2, \dots, w_l, u_1, u_2, \dots, u_h, w_r, \dots, w_t$, where $1 \leq l < r \leq t$. Let y_{max} be the maximum value of y -coordinates of vertices $w_l, w_{l+1}, \dots, w_r$; all these vertices were on $C_o(G_{k-1})$, but all these vertices except w_l and w_r are not on $C_o(G_k)$. (See Fig. 4.) Clearly one must decide $y(U_k) \geq y_{max}$ to obtain a plane drawing of G_k . The algorithm decides $y(U_k)$ to be either y_{max} or $y_{max} + 1$ so that the height H of the drawing becomes as small as possible. There are the following six cases.

Case 1: $y_{max} > y(w_l), y(w_r)$.

In this case, if one decided $y(U_k) = y_{max}$, then G_k could not be a plane drawing. Therefore the algorithm decides $y(U_k) = y_{max} + 1$.

Case 2: $y_{max} = y(w_l) = y(w_r)$.

In this case, if one decided $y(U_k) = y_{max}$, then G_k might not be a plane drawing. Therefore the algorithm decides $y(U_k) = y_{max} + 1$.

Case 3: $y_{max} = y(w_l) > y(w_r)$, and F has a directed edge (w_l, u_1) , that is, $x(w_l) = x(u_1)$.

In this case, if one decided $y(U_k) = y_{max}$, then vertices w_l and u_1 would overlap each other. Therefore the algorithm decides $y(U_k) = y_{max} + 1$.

Case 4: $y_{max} = y(w_l) > y(w_r)$, and F does not have a directed edge (w_l, u_1) , that is, $x(w_l) < x(u_1)$.

In this case, the algorithm decides $y(U_k) = y_{max}$.

Case 5: $y_{max} = y(w_r) > y(w_l)$, and F has a directed edge (w_r, u_h) , that is, $x(w_r) = x(u_h)$.

In this case, if one decided $y(U_k) = y_{max}$, then vertices w_r and u_h would overlap each other. Therefore the algorithm decides $y(U_k) = y_{max} + 1$.

Case 6: $y_{max} = y(w_r) > y(w_l)$, and F does not have a directed edge (w_r, u_h) , that is, $x(u_h) < x(w_r)$.

In this case, the algorithm decides $y(U_k) = y_{max}$.

4. Rectangular Drawing

A *rectangular drawing* of a plane graph G is a drawing of G in which each vertex is drawn as a point, each edge is drawn as a horizontal or vertical line segment without edge-crossings, and each face is drawn as a rectangle. Figure 5 (b) illustrates a rectangular drawing of the plane graph G in Fig. 5 (a). Not every plane graph has a rectangular drawing. Thomassen [32] obtained a necessary and sufficient condition for a biconnected plane graph of the maximum degree Δ at most three to have a rectangular drawing when four vertices of degree two on the outer face are designated as convex corners of a rectangular drawing of the outer face. Four vertices a, b, c and d of G in Fig. 5 (a) are designated corner vertices, and they are drawn as the four convex corners of the rectangular drawing of the outer face of G in Fig. 5 (b). Rahman *et al.* [25] gave a linear-time algorithm, which uses a divide and conquer approach, to obtain a rectangular drawing of such a plane graph. Recently Rahman *et*

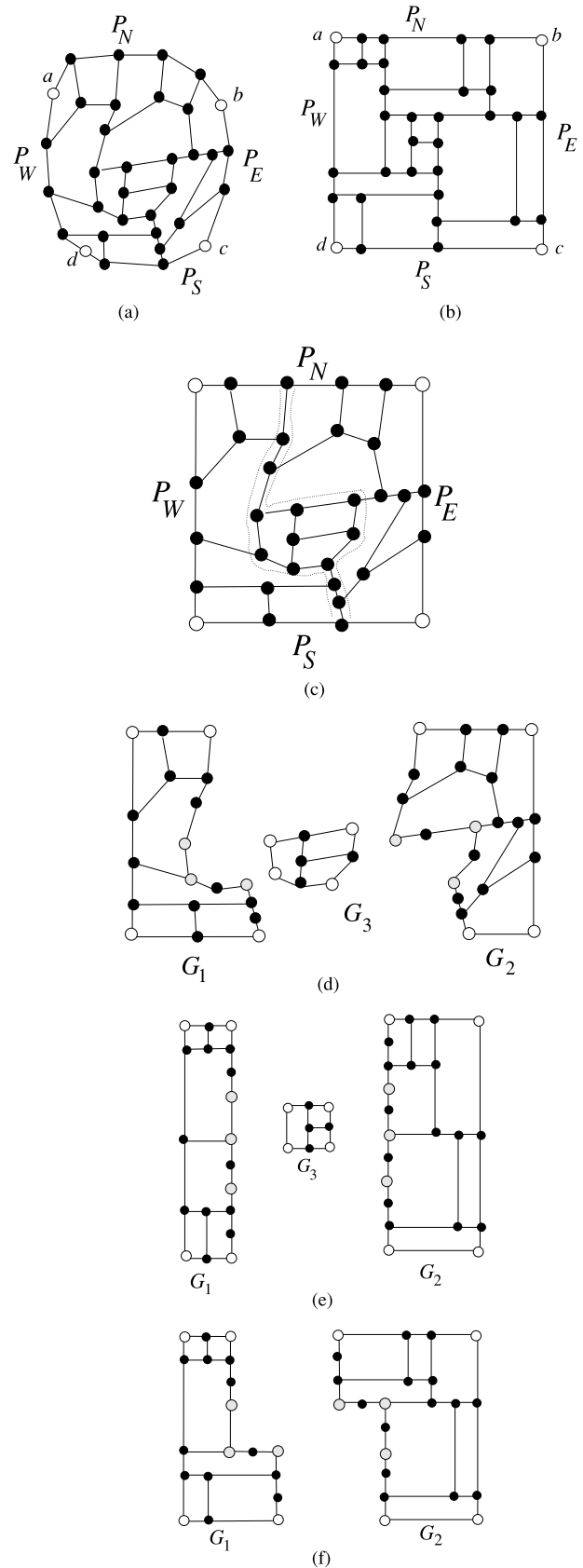


Fig. 5 Illustration of the algorithm for rectangular drawing.

al. [28] gave a necessary and sufficient condition for a plane graph of $\Delta \leq 3$ to have a rectangular drawing for the general case where four vertices are not designated as corners, and developed a linear-time algorithm to find four vertices on the outer face such that there is a rectangular drawing having them as convex corners if they exist. It is still remained as an open problem to obtain an efficient algorithm for finding a rectangular drawing of a plane graph of $\Delta \leq 4$.

On the other hand, Kozminski and Kinnen [18] established a necessary and sufficient condition for the existence of a “rectangular dual” of a plane graph, that is a rectangular drawing of the dual of a given plane graph, and gave an $O(n^2)$ algorithm to obtain it. Based on the characterization of [18], Bhasker and Sahni [1] and He [13] respectively, developed linear-time algorithms to find a rectangular dual of a plane graph. Kant and He [17] gave two more linear-time algorithm. The algorithm in [1] is fairly complicated, and the algorithms in [17] is relatively simpler but need much involvement.

In the rest of this section we present Thomassen’s necessary and sufficient condition [32] and an outline of the algorithm of Rahman *et al.* [25].

We need some definitions. An edge of a plane graph G is called a *leg* of a cycle C if it is incident to exactly one vertex of C and located outside C . A cycle in G is called a *k-legged cycle* of G if C has exactly k legs in G and there is no edge which joins two vertices on C and is located outside C . We are now ready to present Thomassen’s condition as in the following Theorem.

Theorem 4.1: Assume that G is a plane biconnected graph with $\Delta \leq 3$, and that four vertices a, b, c and d of degree 2 on $C_o(G)$ are designated as the four (convex) corners of the outer rectangle. Then G has a rectangular drawing if and only if G satisfies the following two conditions:

- (r1) every 2-legged cycle contains at least two designated vertices, and
- (r2) every 3-legged cycle contains at least one designated vertex.

Consider two examples in Fig. 6, where the four designated corner vertices a, b, c and d are drawn by white circles in each graph. Cycles C_1, C_2 and C_3 are 2-legged, and C_4, C_5 and C_6 are 3-legged. C_3, C_5 and C_6 do not violate the conditions in Lemma 4.1. On the other hand, cycles C_1, C_2

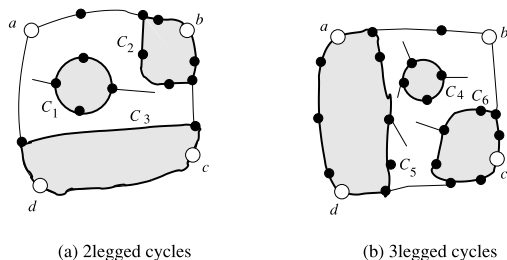


Fig. 6 (a) 2-legged cycles C_1, C_2 and C_3 , and (b) 3-legged cycles C_4, C_5 and C_6 .

and C_4 violate the conditions.

We now give an outline of the algorithm of Rahman *et al.* [25]. Assume that the given plane graph G satisfies Thomassen’s condition in Theorem 4.1. The four designated vertices a, b, c and d divide the outer cycle C_o of G into four paths, the north path P_N , the east path P_E , the south path P_S , and the west path P_W , as illustrated in Fig. 5 (a). The algorithm first fixes the embedding of C_o as a rectangle by drawing the north and south paths on two horizontal straight line segment and the east and west paths on two vertical line segments, as illustrated in Fig. 5 (c). The algorithm then finds either a single “partitioning path” or a pair of “partitioning paths” connecting two opposite sides of the rectangle such that each of the connected subgraphs obtained by splitting G along the partitioning paths satisfies Thomassen’s condition. A pair of partitioning paths of G are indicated by dotted lines in Fig. 5 (c), and the subgraphs G_1, G_2, G_3 obtained by splitting G along the pair of partitioning paths are illustrated in Fig. 5 (d). The four vertices of degree 2 in G_3 drawn by white circles become the corners of a rectangular drawing of G_3 . The vertices of G_1 and G_2 corresponding to them are drawn by gray circles. The algorithm then recursively finds a rectangular drawing of each connected subgraph, as illustrated in Fig. 5 (e). The algorithm finally obtains a rectangular drawing of G by patching the drawings of the subgraphs. Note that the drawings of subgraphs G_1 and G_2 may need to be modified as illustrated in Fig. 5 (f).

5. Box-Rectangular Drawing

A box-rectangular drawing of a plane graph G is a drawing of G such that each vertex is drawn as a (possibly degenerated) rectangle, called a *box*, and the contour of each face is drawn as a rectangle, as illustrated in Fig. 8 (e). If G has multiple edges or a vertex of degree 5 or more, then G has no rectangular drawing but may have a box-rectangular drawing. However, not every plane graph has a box-rectangular drawing. Rahman *et al.* [27] gave a necessary and sufficient condition for a plane graph to have a box-rectangular drawing and gave a linear-time algorithm to find a box-rectangular drawing if it exists. The main idea of their algorithm is to reduce the box-rectangular drawing problem to a rectangular drawing problem. Recently X. He [15] gave an alternate algorithm for finding a box-rectangular drawing of a plane graph if it exists.

In the rest of this section we present the necessary and sufficient condition of Rahman *et al.* [27] and an outline of their algorithm.

Let G be a plane connected graph. For a cycle C in G , we denote by $G(C)$ the plane subgraph of G inside C (including C). We say that cycles C and C' in a plane graph G are *independent* if $G(C)$ and $G(C')$ have no common vertex. A set S of cycles is *independent* if any pair of cycles in S are independent.

Let v be a vertex of degree x in a plane graph G , let $e_1 = vw_1, e_2 = vw_2, \dots, e_x = vw_x$ be the edges incident to v , and assume that these edges $e_1, e_2, \dots, e_x$ appear clockwise

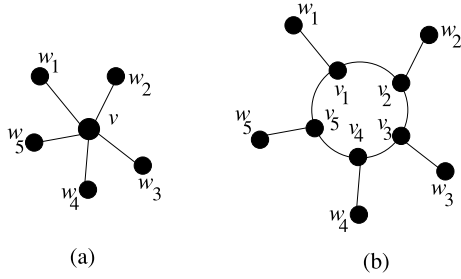


Fig. 7 Replacement of a vertex by a cycle.

around v in this order as illustrated in Fig. 7 (a). Replace v with a cycle $v_1, v_2, \dots, v_x, v_1$, and replace the edges vw_i with v_iw_i for $i = 1, 2, \dots, x$, as illustrated in Fig. 7 (b). We call the operation above the *replacement of a vertex by a cycle*. The cycle $v_1, v_2, \dots, v_x, v_1$ in the resulting graph is called the *replaced cycle* corresponding to vertex v of G . A vertex on a replaced cycle is called a *replaced vertex*, and an edge on a replaced cycle is called a *replaced edge*. We are now ready to present the necessary and sufficient condition of Rahman *et al.* [27] as in the following theorem.

Theorem 5.1: Let G be a plane graph, and let H be the plane graph obtained from G by replacing each vertex of degree four or more by a cycle. Then G has a box-rectangular drawing if and only if H satisfies the following two conditions:

- (br1) every 2-legged or 3-legged cycle in H contains an edge on $C_o(H)$; and
- (br2) $2c_2 + c_3 \leq 4$ for any independent set S of cycles in H , where c_2 and c_3 are the numbers of 2-legged cycles and 3-legged cycles in S , respectively.

We now give an outline of the algorithm of Rahman *et al.* Assume that the given plane graph G satisfies the condition for a box-rectangular drawing in Theorem 5.1. The algorithm determines four vertices of degree 3 as corner vertices on the outer cycle of H , and removes all vertices of degree 2 from H , as illustrated in Figs. 8 (b) and (c). Vertices a, b, c and d are determined as corner vertices in Fig. 8 (b); vertices a and b are original vertices, that is, they are vertices of G , while vertices c and d are replaced vertices. If an original vertex is chosen as a corner vertex, then it is replaced by a cycle and put a dummy vertex of degree 2 on the edge of the cycle on the outer face; a' and b' are such dummy vertices of degree 2 in Fig. 8 (c). If a replaced vertex is chosen, then put a dummy vertex of degree 2 on the replaced edge incident to the vertex which is on the outer face; c' and d' are such dummy vertices of degree 2 in Fig. 8 (c). Let H^* be the resulting graph. Now all the four corner vertices of H^* are dummy vertices of degree 2. In fact, the corner vertices are determined in such a way that H^* satisfies Thomassen's condition for rectangular drawing in Theorem 4.1. The algorithm then finds a rectangular drawing of H^* , as illustrated in Fig. 8 (d), where each replaced cycle is also drawn as a rectangle. Finally converting each rectangle corresponding to a replaced cycle by a box and inserting all removed vertices

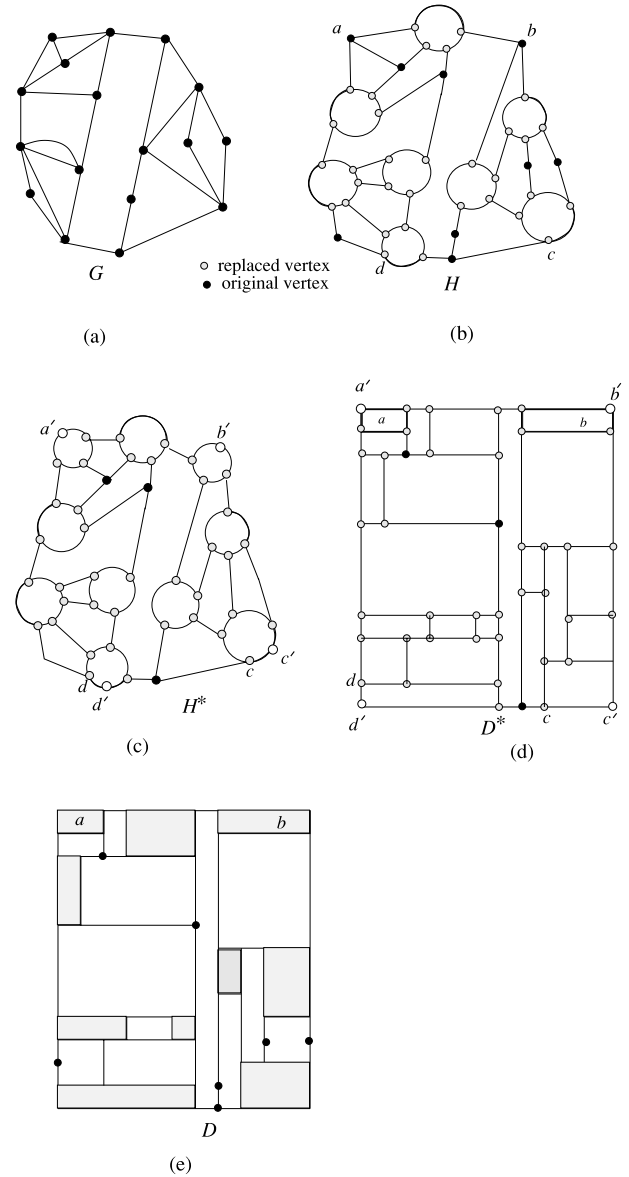


Fig. 8 Illustration of the algorithm for box-rectangular drawing.

of degree 2, the algorithm finally produces a box-rectangular drawing of G , as illustrated in Fig. 8 (e).

6. Orthogonal Drawing

An *orthogonal drawing* of a plane graph G is a drawing of G with the given embedding in which each vertex is mapped to a point, each edge is drawn as a sequence of alternate horizontal and vertical line segments, and any two edges do not cross except at their common end, as illustrated in Fig. 9 (b). Orthogonal drawings have applications in circuit schematics, and it is desired to find an orthogonal drawing with a small number of bends, because a bend corresponds to a “via” or “throughhole,” and increases the fabrication cost of VLSI [19]. It is a challenging problem to find a *bend-minimum* orthogonal drawing, that is, an or-

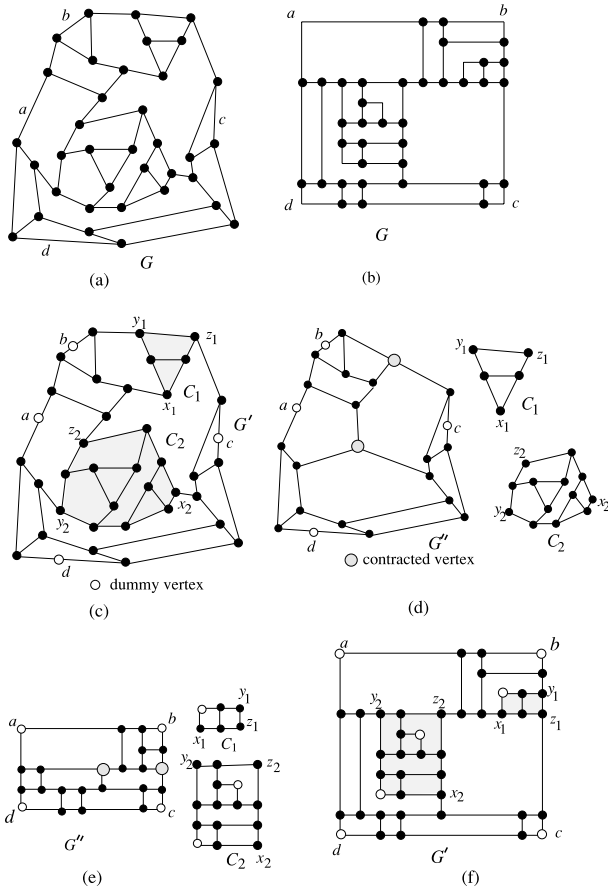


Fig. 9 Illustration of the algorithm for bend-minimum orthogonal drawing.

thogonal drawing with the minimum number of bends [2], [16]. Garg and Tamassia [12] presented an algorithm to find a bend-minimum orthogonal drawing of any plane graph G with $\Delta \leq 4$; if a plane graph has a vertex of degree 5 or more, then there is no orthogonal drawing. They reduce the bend-minimum orthogonal drawing problem to a minimum cost flow problem, and their algorithm [12] takes time $O(n^{7/4} \sqrt{\log n})$ if G has n vertices. Rahman *et al.* [26] presented a linear-time algorithm to find a bend-minimum orthogonal drawing of a triconnected cubic plane graph. The key idea behind their algorithm is to reduce the orthogonal drawing problem to the rectangular drawing problem.

An outline of the algorithm of Rahman *et al.* is illustrated in Fig. 9. Given a plane graph as shown in Fig. 9(a), the algorithm first puts four dummy vertices a, b, c and d of degree two on the outer boundary of G , and let G' be the resulting graph. Figure 9(c) illustrates G' , where the four dummy vertices are drawn by white circles. The algorithm then contracts each of some cycles $C_1, C_2, \dots$ and their insides (shaded in Fig. 9(c)) into a single vertex as shown in Fig. 9(d) so that the resulting graph G'' has a rectangular drawing as shown in Fig. 9(e). The algorithm also finds orthogonal drawings of those cycles $C_1, C_2, \dots$ and their insides recursively (see Figs. 9(d) and (e)). Patching the obtained drawings, the algorithm finds an orthogonal drawing

of G' as shown in Fig. 9(f). Replacing the dummy vertices a, b, c and d in the drawing of G' with bends, the algorithm finally obtains an orthogonal drawing of G as shown in Fig. 9(b).

Rahman and Nishizeki [24] recently gave a linear-time algorithm to find a bend-minimum orthogonal drawing of plane graphs with $\Delta \leq 3$. However, finding a linear-time algorithm to obtain a bend-minimum orthogonal drawing of a plane graph with $\Delta \leq 4$ is still remained as an open problem.

References

- [1] J. Bhasker and S. Sahni, "A linear algorithm to find a rectangular dual of a planar triangulated graph," *Algorithmica*, vol.3, pp.247–278, 1988.
- [2] T.C. Biedl, "Optimal orthogonal drawings of triconnected plane graphs," *Proc. SWAT'96*, LNCS 1097, pp.333–344, 1996.
- [3] N. Chiba, K. Onoguchi, and T. Nishizeki, "Drawing planar graphs nicely," *Acta Inform.*, vol.22, pp.187–201, 1985.
- [4] N. Chiba, T. Yamanouchi, and T. Nishizeki, "Linear algorithms for convex drawings of planar graphs," in *Progress in Graph Theory*, ed. J.A. Bondy and U.S.R. Murty, pp.153–173, Academic Press, 1984.
- [5] M. Chrobak and G. Kant, "Convex grid drawings of 3-connected planar graphs," *Int. J. Comput. Geomet. Appl.*, vol.7, pp.211–223, 1997.
- [6] M. Chrobak and S. Nakano, "Minimum-width grid drawings of plane graphs," *Computational Geometry: Theory and Applications*, vol.10, pp.29–54, 1998.
- [7] M. Chrobak and T. Payne, "A linear-time algorithm for drawing planar graphs on a grid," *Inf. Process. Lett.*, vol.54, pp.241–246, 1995.
- [8] G. Di Battista, P. Eades, R. Tamassia, and I.G. Tollis, "Automatic graph drawing: An annotated bibliography," *Computational Geometry: Theory and Applications*, vol.4, pp.235–282, 1994.
- [9] G. Di Battista, P. Eades, R. Tamassia, and I.G. Tollis, *Graph Drawing*, Prentice Hall, NJ, 1999.
- [10] I. Fáry, "On straight lines representation of plane graphs," *Acta Sci. Math. Szeged*, vol.11, pp.229–233, 1948.
- [11] H. de Fraysseix, J. Pach, and R. Pollack, "How to draw a planar graph on a grid," *Combinatorica*, vol.10, pp.41–51, 1990.
- [12] A. Garg and R. Tamassia, "A new minimum cost flow algorithm with applications to graph drawing," *Proc. Graph Drawing '96*, LNCS 1190, pp.201–226, 1996.
- [13] X. He, "On finding the rectangular duals of planar triangulated graphs," *SIAM J. Comput.*, vol.22, no.6, pp.1218–1226, 1993.
- [14] X. He, "Grid embedding of 4-connected plane graphs," *Discrete & Computational Geometry*, vol.17, pp.339–358, 1997.
- [15] X. He, "A simple linear time algorithm for proper box-rectangular drawings of plane graphs," *J. Algorithms*, vol.40, no.1, pp.82–101, 2001.
- [16] G. Kant, "Drawing planar graphs using the canonical ordering," *Algorithmica*, vol.16, pp.4–32, 1996.
- [17] G. Kant and X. He, "Regular edge labeling of 4-connected plane graphs and its applications in graph drawing problems," *Theor. Comput. Sci.*, vol.172, pp.175–193, 1997.
- [18] K. Kozminski and E. Kinnen, "An algorithm for finding a rectangular dual of a planar graph for use in area planning for VLSI integrated circuits," *Proc. 21st DAC*, pp.655–656, 1984.
- [19] T. Lengauer, *Combinatorial Algorithms for Integrated Circuit Layout*, John Wiley & Sons, Chichester, 1990.
- [20] S. Munemoto, N. Katoh, and G. Imamura, "Finding an optimal floor layout based on an orthogonal graph drawing algorithm," *J. Archit. Plann. Environment Eng. AIJ*, vol.524, pp.279–286, 2000.
- [21] K. Miura, S. Nakano, and T. Nishizeki, "Grid drawings of four-connected plane graphs," *Discrete & Computational Geometry*, vol.26, pp.73–87, 2001.

- [22] K. Miura, S. Nakano, and T. Nishizeki, "Convex grid drawings of four-connected plane graphs," Proc. 11th Annual International Symposium on Algorithms and Computation (ISAAC '00), LNCS 1969, pp.254–265, 2000.
- [23] S. Nakano, M.S. Rahman, and T. Nishizeki, "A linear time algorithm for four partitioning four-connected planar graphs," Inf. Process. Lett., vol.62, pp.315–322, 1997.
- [24] M.S. Rahman and T. Nishizeki, "Bend-minimum orthogonal drawings of plane 3-graphs," Proc. WG '02, LNCS 2573, pp.265–276, 2002.
- [25] M.S. Rahman, S. Nakano, and T. Nishizeki, "Rectangular grid drawings of plane graphs," Comput. Geom. Theory Appl., vol.10, no.3, pp.203–220, 1998.
- [26] M.S. Rahman, S. Nakano, and T. Nishizeki, "A linear algorithm for bend-optimal orthogonal drawings of triconnected cubic plane graphs," J. Graph Alg. Appl., vol.3, no.4, pp.31–62, 1999.
- [27] M.S. Rahman, S. Nakano, and T. Nishizeki, "Box-rectangular drawings of plane graphs," J. Algorithms, vol.37, no.2, pp.363–398, 2000.
- [28] M.S. Rahman, S. Nakano, and T. Nishizeki, "Rectangular drawings of plane graphs without designated corners," Comput. Geom. Theory Appl., vol.21, no.3, pp.121–138, 2002.
- [29] W. Schnyder, "Embedding planar graphs in the grid," Proc. 1st Annual ACM-SIAM Symp. on Discrete Algorithms, pp.138–147, San Francisco, 1990.
- [30] W. Schnyder and W. Trotter, "Convex drawings of planar graphs," Abstracts of the AMS, vol.13, no.5, 92T-05-135, 1992.
- [31] S.K. Stein, "Convex maps," Proc. Amer. Math. Soc., vol.2, pp.464–466, 1951.
- [32] C. Thomassen, "Plane representations of graphs," in Progress in Graph Theory, ed. J.A. Bondy and U.S.R. Murty, pp.43–69, Academic Press Canada, 1984.
- [33] W.T. Tutte, "How to draw a graph," Proc. London Math. Soc., vol.13, pp.743–768, 1963.
- [34] K. Wagner, "Bemerkungen zum vierfarbenproblem," Jahresbericht der Deutschen Mathematiker-Vereinigung, vol.46, pp.26–32, 1936.



Kazuyuki Miura was born in Miyako, Japan in 1973. He received the B.E., M.E., and Ph. D. degrees from Tohoku University, Sendai, Japan, in 1996, 1998 and 2001, respectively. He joined Tohoku University in 2001, and is now a Research Associate of Graduate School of Information Sciences. His research interests are in the field of algorithms for planar graphs, graph drawings and independent spanning trees.



Md. Saidur Rahman was born in Santhia, Pabna, Bangladesh in 1966. He received B. Sc. Engg. degree in Electrical and Electronic Engineering in 1989 and M. Sc. Engg. degree in Computer Science and Engineering in 1992 from Bangladesh University of Engineering and Technology (BUET). He received M. Sc. and Ph. D. degrees in Information Sciences from Tohoku University, Japan in 1996 and 1999 respectively. He joined the department of Computer Science and Engineering, BUET, as a lecturer in

1991 and was appointed as an Assistant Professor in 1999. Currently he is on leave from BUET and working as an Associate Professor at Graduate School of Information Sciences, Tohoku University. His research interests are in the field of graph drawing algorithms and VLSI physical layout algorithms.



Takao Nishizeki was born in Sukagawa, Fukushima, Japan, in 1947. He is currently Professor of algorithm theory in the Graduate School of Information Sciences at Tohoku University in Japan. He received the B.E., M.E. and D.E. degrees in electrical communication engineering from Tohoku University in 1969, 1971 and 1974, respectively. Upon graduation, he joined the faculty at Tohoku University, where in 1976 he was appointed as Associate Professor of the Department of Electrical Commu-

nications, then in 1988 was promoted to Professor. From 1977 to 1978 he was a Visiting Research Mathematician at Carnegie-Mellon University. His research interests are in the field of algorithms for planar graphs, edge-coloring, network flows, VLSI routing and cryptology. He has co-authored two books, edited five books, and published more than 150 technical papers in scientific journals. Nishizeki also founded the International Symposium on Algorithms and Computation (ISAAC) which has been held mainly in Asian-Pacific region annually since 1990. He has been consistently on program committees of international conferences and regularly on editorial boards of several journals including "Algorithmica," "Journal of Graph Algorithms and Applications," "Discrete Math. and Theoretical Computer Science," and "Journal of Combinatorial Optimization." He is a Fellow member of IEEE and ACM.