アルゴリズム 74−2
(2000. 9. 21)

4連結平面グラフの格子凸描画

三浦 一之[1]　中野眞一[2]　西関隆夫[3]

[1,3]東北大学大学院情報科学研究科
〒 980-8579 仙台市青葉区荒巻字青葉05
e-mail: [1]miura@nishizeki.ecei.tohoku.ac.jp [3]nishi@ecei.tohoku.ac.jp
[2]群馬大学工学部情報工学科情報数理第2
〒 376-8515 桐生市天神町1丁目5番1号
e-mail: nakano@cs.gunma-u.ac.jp

あらまし　平面グラフ G の各点を整数格子点上に配置し，各辺が互いに交わることのない直線分となり，各内面が凸多角形となるような描画を G の格子凸描画という．本論文では外周上に4つ以上の点を持つ4連結平面グラフの格子凸描画を見つける線形時間アルゴリズムを与える．アルゴリズムの実行時間は $O(n)$ であり，必要な整数格子の大きさは $W+H \leq n-1$ である．ここで n はグラフの点数であり，W および H はそれぞれ整数格子の幅および高さである．よって格子の面積 $W \times H$ は高々$\lceil (n-1)/2 \rceil \cdot \lfloor (n-1)/2 \rfloor$ である．少なくとも $W+H=n-1$ かつ $W \times H = \lceil (n-1)/2 \rceil \cdot \lfloor (n-1)/2 \rfloor$ の大きさの整数格子が格子凸描画に必要な4連結平面グラフが無限個存在するので，本論文の整数格子の外周の長さおよび面積に関する結果は最適である．

Convex Grid Drawings of Four-Connected Plane Graphs

Kazuyuki Miura[1], Shin-ichi Nakano[2] and Takao Nishizeki[3]

[1,3]Graduate School of Information Sciences,
Tohoku University, Aoba-yama 05, Sendai, 980-8579, Japan.
e-mail: [1]miura@nishizeki.ecei.tohoku.ac.jp [3]nishi@ecei.tohoku.ac.jp
[2]Department of Computer Science, Faculty of Engineering
Gunma University, 1-5-1 Tenjin-cho, Kiryu, Gunma 376-8515, Japan
e-mail: nakano@cs.gunma-u.ac.jp

Abstract　A convex grid drawing of a plane graph G is a drawing of G on the plane so that all vertices of G are put on grid points, all edges are drawn as straight-line segments between their endpoints without any edge-intersection, and every face boundary is a convex polygon. In this paper we give a linear-time algorithm for finding a convex grid drawing of any 4-connected plane graph G with four or more vertices on the outer face boundary. The algorithm yields a drawing in an integer grid such that $W+H \leq n-1$ if G has n vertices, where W is the width and H is the height of the grid. Thus the area $W \times H$ of the grid is at most $\lceil (n-1)/2 \rceil \cdot \lfloor (n-1)/2 \rfloor$. Our bounds on the grid sizes are optimal in the sense that there exist an infinite number of 4-connected plane graphs whose convex drawings need grids such that $W+H=n-1$ and $W \times H = \lceil (n-1)/2 \rceil \cdot \lfloor (n-1)/2 \rfloor$.

1 Introduction

Recently automatic aesthetic drawing of graphs have created intense interest due to their broad applications, and as a consequence, a number of drawing methods have appeared [BETT99, CK97, CN98, CON85, CP95, CYN84, Fa48, FPP90, He97, Ka96, Sc90, Tu63]. In this paper, we deal with the "convex grid drawing" of a plane graph. Throughout the paper we denote by n the number of vertices of a graph G. The $W \times H$ *integer grid* consists of $W+1$ vertical grid lines and $H+1$ horizontal grid lines, and has a rectangular contour. W and H are called the *width* and *height* of the integer grid, respectively.

The most typical drawing of a plane graph G is *the straight line drawing* in which all vertices of G are drawn as points and all edges are drawn as straight line segments without any edge-intersection. A straight line drawing of G is called a *grid drawing* of G if the vertices of G are put on grid points of integer coordinates. Every plane graph has a grid drawing on an $(n-2) \times (n-2)$ grid [BETT99, CP95, FPP90, Sc90]. A straight line drawing of a plane graph G is often aesthetically pretty if every face boundary is drawn as a convex polygon [CON85, Tu63]. 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 [Tu63], and such a grid drawing can be found in linear time [CON85, CYN84]. 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 such a grid drawing can be found in linear time [CK97, ST92]. 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.

In this paper we give an answer to this problem. That is, we give an algorithm which finds in linear time a convex grid drawing of any given 4-connected plane graph G 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 Figs. 1 and 5(d). The assumption that a given plane graph has four or more vertices on the outer face boundary does not lose much generality, because any 4-connected plane graph has at least three vertices on the outer face boundary. Our bounds on $W + H$ and $W \times H$ are optimal in the 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 a rectangular grid with $W + H \leq n$ [He97] and on an almost square grid with $W = \lceil (n-1)/2 \rceil$ and $H = \lceil (n-1)/2 \rceil$ [MNN99], but the the drawing is not always convex.

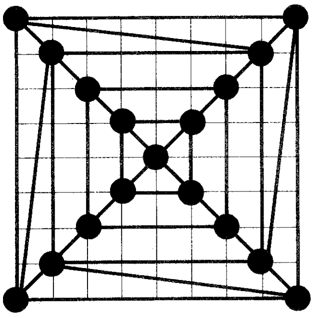

Figure 1: Nested quadrangles attaining our bounds.

The remainder of the paper is organized as follows. In Section 2, we give some definitions and a lemma. In Section 3, we give our linear-time algorithm. In Section 4, we prove that the algorithm finds a convex grid drawing of G. In Section 5, we prove that $W + H \leq n - 1$. Finally we conclude in Section 6.

2 Preliminaries

In this section we introduce some definitions and a lemma.

Let $G = (V, E)$ be a simple connected undirected graph having no multiple edge or loop. V is the vertex set, and E is the edge set of G. Let $x(v)$ and $y(v)$ be the x- and y-coordinates of vertex $v \in V$, respectively. An edge joining vertices u and v is denoted by (u, v). The *degree* of a vertex v in G is the number of neighbors of v in G, and is denoted by $d(v, G)$. The *connectivity* $\kappa(G)$ of a graph G is the minimum number of vertices whose removal results in a disconnected graph or a single-vertex graph K_1. A graph G is *k-connected* if $\kappa(G) \geq k$.

A graph is *planar* if it can be embedded in the plane so that no two edges intersect geometrically except at a vertex to which they are both incident. A *plane graph* is a planar graph with a fixed embedding. A plane graph divides the plane into connected regions called *faces*. We denote the boundary of a face by a clockwise sequence of the vertices on the boundary. We call the boundary of the outer face of a plane graph G *the contour* of G, and denote it by $C_o(G)$.

The "4-canonical decomposition" of a plane graph G [NRN97] playing a crucial role in our 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 [Ka96]; and the "4-canonical ordering," which is used to find a "visibility representation" and a grid drawing of a 4-connected plane graph [He97, KH97, MNN99]. A 4-canonical decomposition $\Pi = (U_1, U_2, \cdots, U_{12})$ is illustrated in Fig. 2 for a 4-connected plane graph. Let m be a natural number, and let $\Pi = (U_1, U_2, \cdots, U_m)$ be a partition of set V to m subsets $U_1, U_2, \cdots, U_m$ of V where $U_1 \bigcup U_2 \bigcup \cdots \bigcup U_m = V$ and $U_i \bigcap U_j = \phi$ 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 $U_1 \bigcup$

$U_2 \bigcup \cdots \bigcup U_k$, and let $\overline{G_k}$ be the plane subgraph of G induced by the vertices in $U_{k+1} \bigcup U_{k+2} \bigcup \cdots \bigcup U_m$. 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:

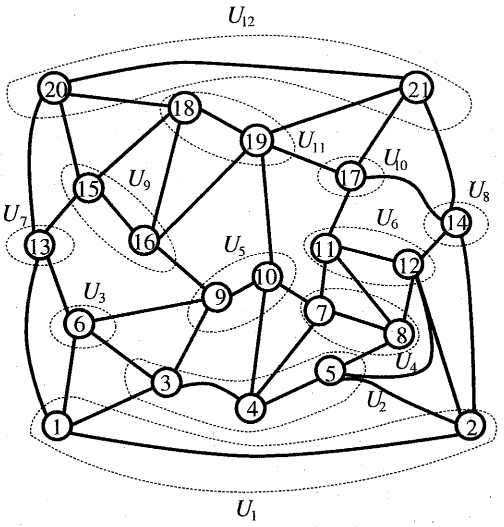

Figure 2: A 4-canonical decomposition of a 4-connected plane graph having $n = 21$ vertices.

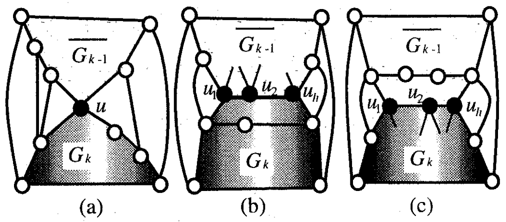

Figure 3: Illustration for the three conditions (a)–(c) of (co3).

(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, $2 \le k \le m-1$, both G_k and $\overline{G_{k-1}}$ are biconnected (in Fig. 3 G_k is darkly shaded, and $\overline{G_{k-1}}$ is lightly shaded); and

(co3) for each k, $2 \le k \le m-1$, one of the following three conditions holds (the vertices in U_k are drawn by black circles in Fig. 3):

(a) U_k is a singleton set of a vertex u on $C_o(G_k)$ such that $d(u, G_k) \ge 2$ and $d(u, \overline{G_{k-1}}) \ge 2$ (see Fig. 3(a)).

(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 $d(u, \overline{G_{k-1}}) \ge 3$ for each vertex $u \in U_k$(see Fig. 3(b)).

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

Although the definition of a 4-canonical decomposition above is slightly different from that in [NRN97], they are effectively equivalent to each other. The following lemma is known.

Lemma 2.1 [NRN97] Let G be a 4-connected plane graph having at least four vertices on $C_o(G)$. Then G has a 4-canonical decomposition Π, and Π can be found in linear time.

By the condition (co3), one may assume that for each k, $1 \le k \le m$, the vertices in U_k consecutively appear clockwise on $C_o(G_k)$. However, the clockwise order on $C_o(G_1)$ is not well-defined since $G_1 = K_2$. So we assume that the two vertices in U_1 consecutively appear counterclockwise on $C_o(G)$ as illustrated in Fig. 2. We number all vertices of G by $1, 2, \cdots, n$ so that they appear in $U_1, U_2, \cdots, U_m$ in this order, and call each vertex in G by the number i, $1 \le i \le n$. Thus one can define an order $<$ among the vertices in G.

In the remainder of this section, we define some terms which are used in our algorithm. The *lower neighbor* of u is the neighbors of u which are smaller than u. The *upper neighbor* of u is the neighbors of u which are larger than u. Every upper neighbor v of any vertex u satisfies $y(v) \ge y(u)$ in our 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 $d_{low}(u, G) \ge 1$, and every vertex u except vertex n satisfies $d_{up}(u, G) \ge 1$. Let $2 \le k \le m-1$ and $U_k = \{u_1, u_2, \cdots, u_h\}$. If U_k satisfies the condition (co3)(a), then $h = 1$ and $d_{low}(u_1), d_{up}(u_1) \ge 2$. If U_k satisfies condition (co3)(b), then $d_{low}(u_i) = 1$ for each u_i, $1 \le i \le h-1$, $d_{low}(u_h) = 2$, $d_{up}(u_i) \ge 3$ for each u_i, $1 \le i \le h-1$, and $d_{up}(u_h) \ge 2$. If U_k satisfies condition (co3)(c), then $d_{low}(u_1) \ge 2$, $d_{low}(u_i) \ge 3$ for each u_i,

$2 \le i \le h$, $d_{up}(u_1) = 2$, and $d_{up}(u_i) = 1$ for each u_i, $2 \le i \le h$. We denote by $w_m(u)$ the largest neighbor of u, $1 \le u \le n-1$. 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 $d_{out}(u, D)$.

3 Algorithm

In this section, we present our algorithm which finds a convex grid drawing of any given 4-connected plane graph G with four or more vertices on the contour $C_o(G)$. Our algorithm determines only the integer coordinates of the vertices $1, 2, \cdots, n$ of G effectively in this order. One can immediately find a (straight line) grid drawing of G from the coordinates. We first determine the x-coordinates of all vertices, and then determine the y-coordinates.

3.1 How to compute x-coordinates

We first show how to compute the x-coordinates of all vertices. Our algorithm 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, we put an upper neighbor j of i on the same vertical grid line as i, that is, we determine $x(j) = x(i)$ and hence $y(j) > y(i)$ of course, as illustrated in Fig. 4. We wish to choose as j the largest neighbor $w_m(i)$ of i (this is crucial for making every face boundary a convex polygon). However, it is impossible for a case where $w_m(i)$ 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, $w_m(i') = w_m(i)$. 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 we put the largest one j among them on the same vertical grid line as i. If there dose not exist such an upper neighbor of i, then we do 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     $E_F := \phi$ ; {the initial forest $F = (V, \phi)$ 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, $E_F := E_F \bigcup \{(i, j)\}$;
  end.
```

Since $d_{in}(i, F), d_{out}(i, F) \le 1$ for each vertex i, $1 \le i \le n$, F is a forest and each component of F is a directed path. Clearly $d_{in}(1, F) = d_{in}(2, F) = 0$ and $d_{out}(n-1, F) = d_{out}(n, F) = 0$. Fig. 5(b) illustrates the directed forest F of the graph G in Fig. 5(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. 5(d). 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. That is, we determine a total order among all starting vertices of paths in F. For this purpose, using the following **procedure** Total-Order, we find a directed path P going from vertex 1 to vertex 2 passing through all starting vertices of F. In Fig. 5(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 $w_m(i)$;
4            let $j'$ be the starting vertex of the path in $F$ containing vertex $j$; {$2 \ne j' < i$}
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 $j'$ as illustrated in Fig. 6(a)}
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 $k$ as illustrated in Fig. 6(b)}
    end
end.
```

We construct a weighted tree T rooted at 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. 5(c).) Then the x-coordinate $x(i)$ of each vertex i, $1 \le i \le n$, is the length of the path from root 1 to i in T. Thus $x(1) = 0$, and the width $W = x(2)$ of our 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. Thus one may regard that a vertex of in-degree 0 in F except vertex 1 increases W by one.

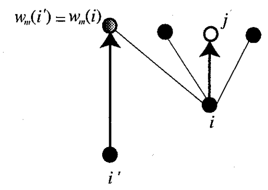

Figure 4: Selection of an i's upper neighbor j above i.

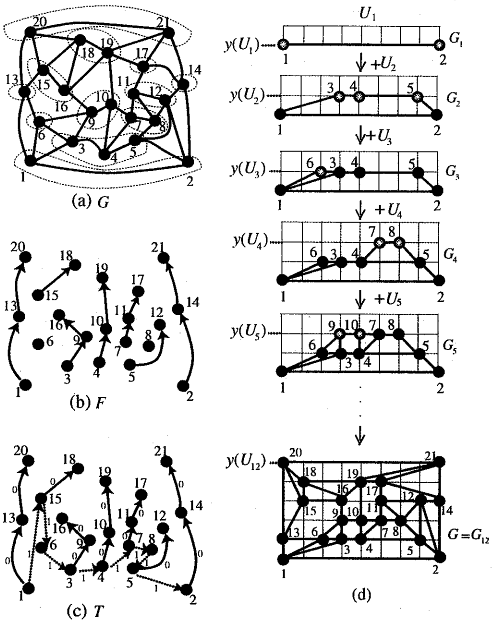

Figure 5: Illustration of our algorithm.

3.2 How to compute y-coordinates

We now show how to compute y-coordinates. For each k, $1 \le k \le m$, y-coordinates of all vertices in $U_k = \{u_1, u_2, \cdots, u_h\}$ are determined as the same integer, which is denoted by $y(U_k)$. Thus the path $u_1, u_2, \cdots, u_h$ 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. 5(d).) Furthermore, we determine the y-coordinates $y(U_1), y(U_2), \cdots, y(U_m)$ in this order. Thus $H = y(U_m)$.

We first determine the y-coordinate $y(U_1)$ of $U_1 = \{1, 2\}$ as $y(U_k) = 0$. Thus we draw $G_1 = K_2$ as a horizontal line segment connecting points $(x(1), 0)$ and $(x(2), 0)$, as illustrated in the top figure of Fig. 5(d).

Suppose that $y(U_1), y(U_2), \cdots, y(U_{k-1})$, $k \ge 2$, have already been determined, that is, G_{k-1} has already been drawn, and we are now going to determine $y(U_k)$ and obtain a drawing of G_k by adding the vertices in U_k to the drawing of G_{k-1}. Let $C_o(G_{k-1}) = w_1, w_2, \cdots, w_t$, where $w_1 = 1$ and $w_t = 2$. Let $C_o(G_k) = w_1, w_2, \cdots, w_l, u_1, u_2, \cdots, u_h, w_r, \cdots, w_t$, where $1 \le l < r \le t$. Let y_{max} be the maximum value of y-coordinates of vertices $w_l, w_{l+1}, \cdots, w_r$; 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. 7.) Clearly we must determine $y(U_k) \ge y_{max}$ to obtain a plane drawing of G_k. Our algorithm determines $y(U_k)$ 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: $y_{max} > y(w_l), y(w_r)$. (See Fig. 7(a).)

In this case, if we determined $y(U_k) = y_{max}$, then G_k could not be a plane drawing. Therefore we determine $y(U_k) = y_{max} + 1$.

Case 2: $y_{max} = y(w_l) = y(w_r)$. (See Fig. 7(b).)

In this case, if we determined $y(U_k) = y_{max}$, then G_k might not be a plane drawing. Therefore we determine $y(U_k) = y_{max} + 1$.

Case 3: $y_{max} = y(w_l) > y(w_r)$, and F has a directed edge (w_l, u_1), that is, $x(w_l) = x(u_1)$. (See Fig. 7(c).)

In this case, if we determined $y(U_k) = y_{max}$, then vertices w_l and u_1 would overlap each other. Therefore we determine $y(U_k) = y_{max} + 1$.

Case 4: $y_{max} = y(w_l) > y(w_r)$, and F does not have a directed edge (w_l, u_1), that is, $x(w_l) < x(u_1)$. (See Fig. 7(d).)

In this case, we determine $y(U_k) = y_{max}$.

Case 5: $y_{max} = y(w_r) > y(w_l)$, and F has a directed edge (w_r, u_h), that is, $x(w_r) = x(u_h)$. (See Fig. 7(e).)

In this case, if we determined $y(U_k) = y_{max}$, then vertices w_r and u_h would overlap each other. Therefore we determine $y(U_k) = y_{max} + 1$.

Case 6: $y_{max} = y(w_r) > y(w_l)$, and F does not have a directed edge (w_r, u_h), that is, $x(u_h) < x(w_r)$. (See Fig. 7(f).)

In this case, we determine $y(U_k) = y_{max}$.

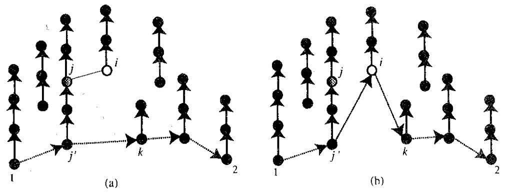

Figure 6: Illustration of Total-Order.

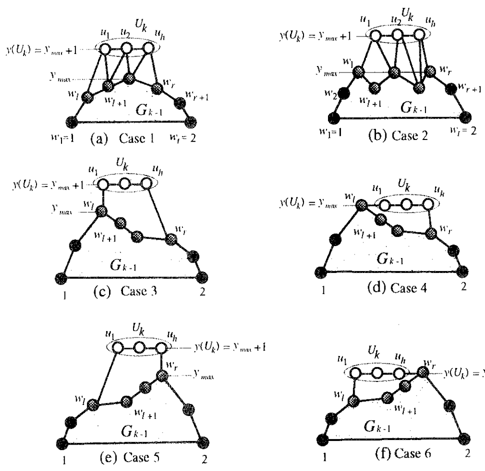

Figure 7: Illustration for the six cases.

We then have the following theorem.

Theorem 1 Our algorithm takes linear time.

Proof By Lemma 2.1, a 4-canonical decomposition can be found in linear time. Clearly the forest F and the rooted tree T can be found in linear time, and the x-coordinates of vertices can be found from T in linear time. Furthermore, the y-coordinates can be found in linear time as above. Thus our algorithm runs in linear time. Q.E.D

4 Proof for convex grid drawing

In this section, we prove that our algorithm finds a convex grid drawing of G. Since clearly every vertex has integer coordinates, it suffices to show that the drawing obtained by our algorithm is a convex drawing.

If $U_k = \{u_1, u_2, \cdots, u_h\}$ satisfies the condition (co3)(b), then for each i, $2 \leq i \leq h-1$, $d_{in}(u_i, F) = 0$ and hence u_i is a starting vertex of a directed path of F. Similarly, if U_k satisfies the condition (co3)(c), then for each i, $2 \leq i \leq h-1$, $d_{out}(u_i, F) = 0$ and hence u_i is an ending vertex of a directed path of F. We thus have the following lemma.

Lemma 4.1 Let $2 \leq k \leq m$, let $U_k = \{u_1, u_2, \cdots, u_h\}$, and let $C_o(G_k) = w_1, w_2, \cdots, w_l, u_1, u_2, \cdots, u_h, w_r, \cdots, w_t$, where $w_1 = 1$, $w_t = 2$, and $1 \leq l < r \leq t$. Then $x(w_l) \leq x(u_1) < x(u_2) < \cdots < x(u_h) \leq w_r$.

Since $y(U_k)$ is equal to either y_{max} or $y_{max} + 1$, the following lemma clearly holds.

Lemma 4.2 If vertices u and v are adjacent and $u < v$, then $y(u) \leq y(v)$.

Our algorithm finds the drawing of $G_1, G_2, \cdots, G_m (= G)$ in this order, as illustrated in Fig. 5(d). Thus, assuming that the drawing of G_{k-1}, $k \geq 2$, is convex, we shall show that the drawing of G_k is convex. However, it is difficult to show that the drawing of G_k is convex for the case where either $k = m$ or U_k, $2 \leq k \leq m-1$, satisfies the condition (co3)(c). Therefore, subdividing all such sets U_k, we obtain another partition of V as follows. Let $\Pi = (U_1, U_2, \cdots, U_m)$ be a 4-canonical decomposition of G. For each U_k such that either $k = m$ or U_k satisfies the condition (co3)(c), let $U_k = \{u_1, u_2, \cdots, u_{l_k}\}$ and replace U_k in Π with singleton sets $\{u_1\}, \{u_2\}, \cdots, \{u_{l_k}\}$. We call the resulting partition $\Pi' = (U_1, U_2^1, U_2^2, \cdots, U_2^{l_2}, U_3^1, U_3^2, \cdots, U_3^{l_3}, \cdots, U_m^1, U_m^2)$ of V a *refined decomposition* of G. If either $k = m$ or U_k satisfies the condition (co3)(c), then $U_k = U_k^1 \bigcup U_k^2 \bigcup \cdots \bigcup U_k^{l_k}$, $l_k = |U_k|$ and $|U_k^i| = 1$ for each i, $1 \leq i \leq l_k$. Otherwise, $l_k = 1$ and $U_k = U_k^1$.

For each k, $2 \leq k \leq m$, and for each i, $1 \leq i \leq l_k$, we denote by G_k^i the plane subgraph of G induced by the vertices in $U_1 \bigcup U_2 \bigcup \cdots \bigcup U_{k-1} \bigcup U_k^1 \bigcup U_k^2 \bigcup \cdots \bigcup U_k^i$. Moreover, for each k, $2 \leq k \leq m$, and for each i, $0 \leq i \leq l_k - 1$, we denote by $\overline{G_k^i}$ the plane subgraph of G induced by the vertices in $U_k^{i+1} \bigcup U_k^{i+2} \bigcup \cdots \bigcup U_k^{l_k} \bigcup U_{k+1} \bigcup \cdots \bigcup U_m$. For notational convenience, let $G_k^0 = G_{k-1}$ and $\overline{G_k^{l_k}} = \overline{G_k}$.

Let $k \geq 2$ and $U_k^i = \{u_1, u_2, \cdots, u_h\}$. By the definition of a refined decomposition, vertices $u_1, u_2, \cdots, u_h$ consecutively appear clockwise on $C_o(G_k^i)$ in this order. Let $C_o(G_k^{i-1}) = w_1, w_2, \cdots, w_t$, where $w_1 = 1$ and $w_t = 2$. Let $C_o(G_k^i) = w_1, w_2, \cdots, w_l, u_1, u_2, \cdots, u_h, w_r, \cdots, w_t$, where $1 \leq l < r \leq t$. We call w_l the *left leg* of U_k^i, and w_r the *right leg* of U_k^i. By the definition of a 4-canonical decomposition and a refined decomposition, the left leg of U_k^i is different from the right leg of U_k^i.

We now have the following lemma for the drawing of G_k^i.

Lemma 4.3 For each k, $2 \leq k \leq m$, and each i, $0 \leq i \leq l_k$, the following (i)-(iii) hold:

(i) the path going clockwise on $C_o(G_k^{i-1})$ from vertex $w_1 = 1$ to vertex $w_t = 2$ is "x-monotone," that is, $x(w_1) \leq x(w_2) \leq \cdots \leq x(w_t)$ (such a path is drawn by thick solid lines in Fig. 8);

(ii) the path going clockwise on $C_o(G_k^{i-1})$ from w_l to w_r is "quasi-convex," that is, there is no vertex w_p such that $l < p < r$ and $y(w_{p-1}) < y(w_p) > y(w_{p+1})$ (all vertices in such a path are drawn by gray circles in Fig. 8), and $w_l, w_{l+1}, \cdots, w_r, w_l$ is a convex polygon in particular if U_k satisfies the condition (co3)(b) (as illustrated in Fig. 8(a)); and

(iii) if a vertex v on $C_o(G_k^{i-1})$ is an inner vertex of G, that is, v is not on $C_o(G)$, and the interior angle of the polygon $C_o(G_k^{i-1})$ at vertex v is less than $180°$, then v has at least one neighbor in $\overline{G_k^{i-1}}$ (the edges joining v and such neighbors are drawn by thin dotted line in Fig. 8).

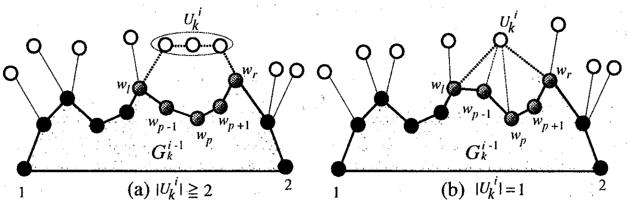

Figure 8: Illustration for Lemma 4.3.

Proof Investigating the algorithm in detail and using Lemmas 4.1 and 4.2, one can prove the lemma. The detail is omitted in this extended abstract due to the page limitation. Q.E.D

Using Lemma 4.3, one can prove that our algorithm obtains a convex grid drawing. Note that all inner face boundaries newly formed in G_k^i are convex polygons as illustrated in Fig. 8 (all such faces are not shaded in Fig. 8).

5 Grid size

In this section, we prove the following theorem.

Theorem 2 Our algorithm finds a convex grid drawing of G on a grid such that $W + H \leq n - 1$.

Proof Due to the page limitation, we outline a proof in this extended abstract. We denote the height of the drawing of G_k^i by $H(G_k^i)$. We say that a vertex $u_j \in U_k^i$, $3 \leq u_j \leq n$, is *increasing* if $d_{in}(u_j, F) = 0$ and $H(G_k^i) = H(G_k^{i-1}) + 1$, that is, u_j increases $W + H$ by two. We say that a vertex $u_j \in U_k^i$ is *preserving* if $d_{in}(u_j, F) \neq 0$ and $H(G_k^i) = H(G_k^{i-1})$, that is, u_j preserves $W + H$. In particular, we say that vertex 1 is preserving, since the graph consisting of only vertex 1 can be drawn on a 0×0 grid with $W + H = 0$. Vertex 2 is neither increasing nor preserving, since G_1 can be drawn on a 1×0 grid with $W + H = 1$ and hence vertex 2 increases $W + H$ by one. Let I be the set of all increasing vertices in G, and let P be the set of all preserving vertices in G. Then each of the vertices in $V - P - I$ increases $W + H$ by one. Therefore $W + H = 2|I| + (n - |I| - |P|) = n + |I| - |P|$. Investigating the method for deciding x- and y-coordinates of vertices in detail, one can prove that for each increasing vertex v there is at least one preserving vertex (other than vertex 1) around v and all these preserving vertices are distinct from each other. For example, if u_1 is increasing in Case 1 illustrated in Fig. 7(a), then u_2 is preserving. (The detail is omitted in this extended abstract.) Thus we have $|P| \geq |I| + 1$, and hence $W + H \leq n - 1$. Q.E.D

6 Conclusion

In this paper we gave an algorithm which finds in linear time a convex grid drawing of any given 4-connected plane graph G on an integer grid such that $W + H \leq n - 1$ if G has four or more vertices on the outer face boundary. Since $W + H \leq n - 1$, the area $W \times H$ of the grid is at most $\lceil (n-1)/2 \rceil \cdot \lfloor (n-1)/2 \rfloor$. Our bounds on $W + H$ and $W \times H$ are optimal in the sense that there exist an infinite number of 4-connected plane graphs which need grids such that $W + H = n + 1$ and $W \times H = \lceil (n-1)/2 \rceil \cdot \lfloor (n-1)/2 \rfloor$. Our convex grid drawing algorithm uses a 4-canonical decomposition, and is very simple. On the other hand the previously known algorithms to find a grid drawing of a 4-connected plane graph use a 4-canonical ordering, and do not always find a convex drawing [He97, MNN99].

References

[BETT99] G. Di Battista, P. Eades, R. Tamassia and I.G. Tollis, *Graph Drawing*, Prentice Hall, NJ (1999).

[CK97] M. Chrobak and G. Kant, *Convex grid drawings of 3-connected planar graphs*, International Journal of Computational Geometry and Applications, 7, 211-223 (1997).

[CN98] M. Chrobak and S. Nakano, *Minimum-width grid drawings of plane graphs*, Computational Geometry: Theory and Applications, 10, 29-54 (1998).

[CON85] N. Chiba, K. Onoguchi and T. Nishizeki, *Drawing planar graphs nicely*, Acta Inform., 22, 187-201 (1985).

[CP95] M. Chrobak and T. Payne, *A linear-time algorithm for drawing planar graphs on a grid*, Information Processing Letters, 54, 241-246 (1995).

[CYN84] N. Chiba, 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, 153-173 (1984).

[Fa48] I. Fáry, *On straight lines representation of plane graphs*, Acta. Sci. Math. Szeged, 11, 229-233 (1948).

[FPP90] H. de Fraysseix, J. Pach and R. Pollack, *How to draw a planar graph on a grid*, Combinatorica, 10, 41-51 (1990).

[He97] X. He, *Grid embedding of 4-connected plane graphs*, Discrete & Computational Geometry, 17, 339-358 (1997).

[Ka96] G. Kant, *Drawing planar graphs using the canonical ordering*, Algorithmica, 16, 4-32 (1996).

[KH97] G. Kant and X. He, *Regular edge labeling of 4-connected plane graphs and its applications in graph drawing problems*, Theoretical Computer Science, 172, 175-193 (1997).

[MNN99] K. Miura, S. Nakano and T. Nishizeki, *Grid drawings of four-connected plane graphs*, Proc. Graph Drawing'99 (GD'99), LNCS 1731, 145-154 (1999).

[NRN97] S. Nakano, M. Saidur Rahman and T. Nishizeki, *A linear time algorithm for four partitioning four-connected planar graphs*, Information Processing Letters, 62, 315-322 (1997).

[Sc90] W. Schnyder, *Embedding planar graphs in the grid*, Proc. 1st Annual ACM-SIAM Symp. on Discrete Algorithms, San Francisco, 138-147 (1990).

[ST92] W. Schnyder and W. Trotter, *Convex drawings of planar graphs*, Abstracts of the AMS, 13, 5, 92T-05-135 (1992).

[Tu63] W.T. Tutte, *How to draw a graph*, Proc. London Math. Soc., 13, 743-768 (1963).