Grid Embedding of 4-Connected Plane Graphs

Xin He

Department of Computer Science, State University of New York at Buffalo, Buffalo, NY 14260. The work was partially supported by NSF grant CCR-9205982.

Abstract. A straight line grid embedding of a plane graph G is a drawing of G such that the vertices are drawn at grid points and the edges are drawn as non-intersecting straight line segments. In this paper, we show that, if a 4-connected plane graph G has at least 4 vertices on its exterior face, then G can be embedded on a grid of size $W \times H$ such that $W + H \leq n$, $W \leq (n+3)/2$ and $H \leq 2(n-1)/3$, where n is the number of vertices of G. Such an embedding can be computed in linear time.

1 Introduction

A *straight line grid embedding* of a n-vertex planar graph is a drawing where the vertices are located at distinct grid points, and the edges are represented by straight line segments. Such embeddings on reasonably small grids are very useful in visualizing planar graphs on graphic screens [5]. Wagner [19], Fáry [6], and Stein [17] independently showed that every planar graph has a straight line embedding. Since then, many embedding algorithms have been reported (e.g. [18, 1]). The earlier algorithms all suffer two serious drawbacks, as noted in [7]. First, they require high-precision real arithmetic, and therefore cannot be used even for a graph of moderate size. Second, in the drawings produced by these algorithms, the ratio of the largest distance to the smallest distance between vertices are so large (exponential in n) that it is very difficult to view those drawings on graphic screens. In view of these drawbacks, Rosenstiehl and Tarjan [12] posed the problem of computing a straight line embedding on a grid of polynomial size. Schnyder [14] proved that every planar graph has a straight line embedding on a $(2n-4) \times (2n-4)$ grid. Independently, de Fraysseix, Pach, and Pollack showed that every planar graph has an embedding on a $(2n-4) \times (n-2)$ grid, which can be computed in $O(n \log n)$ time [7]. The running time of their algorithm was improved to $O(n)$ [4]. Schnyder proved the existence of an embedding on an $(n-2) \times (n-2)$ grid [13] and gave an $O(n)$ time algorithm to compute such an embedding [15]. Schnyder's algorithm can be implemented in parallel in $O(\log n \log\log n)$ time with $n/\log n$ processors on a PRAM [8]. It was shown in [9] that every 3-connected planar graph G can be embedded on a $(2n-4) \times (n-2)$ grid such that all internal faces of G are convex. The grid size of such embedding is reduced to $(n-2) \times (n-2)$ in [2, 16].

There exists a plane graph G such that, for any straight line grid embedding of G, each dimension of the grid needs to be at least $\lfloor 2(n-1)/3 \rfloor$ even if the other dimension is allowed to be unbounded [7, 3]. It has been conjectured that every planar graph can be embedded on a $2n/3 \times 2n/3$ grid. This conjecture remains open. Chrobak and Nakano showed that every planar graph has a straight line embedding on a $(\lfloor 2(n-1)/3 \rfloor) \times (4\lfloor 2(n-1)/3 \rfloor - 1)$ grid [3].

For the grid embedding problem, we can assume that all internal faces of G are triangles. (If not, we can triangulate G and remove the added edges after an embedding is obtained.) If all internal faces of G are triangles, it is an *internally triangulated plane* graph. If the external face of G is also a triangle, then G is a *triangulated plane* graph. A *non-empty triangle* of G is a triangle containing some vertices in its interior. In this paper we show that if G is 4-connected and has at least 4 vertices on its external face, then the above mentioned $(2n/3 - 1) \times (2n/3 - 1)$ lower bound on grid size does not hold. We will prove:

Theorem 1. *Every n vertex 4-connected plane graph G with at least 4 vertices on the external face has a straight line embedding on a $W \times H$ grid such that $W + H \leq n$, $W \leq (n+3)/2$ and $H \leq 2(n-1)/3$. Such an embedding can be computed in linear time.*

In Theorem 1, we assume G is given by its adjacency list representation, where the neighbors of each vertex are given in clockwise order of embedding. The 4-connectivity of G can be checked in linear time [10]. Every such a graph G can be internally triangulated so that it has no non-empty triangles. From now on, we only consider such internally triangulated plane graphs. The present paper is organized as follows. In Section 2, we review some definitions and describe a *generic shift algorithm* in [3], which is the basis of our algorithm. In Section 3, we present our algorithm. In section 4, we prove Theorem 1.

2 Preliminaries

The embedding algorithms in [7, 4, 3, 9] are based on the following concept [7].

Definition 2. Let $G = (V, E)$ be a triangulated plane graph and $\pi = v_1, v_2 \ldots v_n$ an ordering of V such that the edge (v_1, v_2) is on the external face of G. Let G_k be the subgraph of G induced by $v_1, \ldots, v_k$ and C_k be the external face of G_k. π is called a *canonical ordering* of G if the following hold for $k = 3, \ldots, n$:
(co1) G_k is 2-connected and all internal faces of G_k are triangles.
(co2) C_k contains the edge (v_1, v_2).
(co3) If $k < n$, then v_{k+1} is in the external face of G_k and all neighbors of v_{k+1} in G_k belongs to C_k.
(co4) If $k < n$, then v_k has at least one neighbor v_j with $j > k$.

Every triangulated plane graph has a canonical ordering [7]. We will use $\prec$ to denote the linear order of the canonical ordering. Fig 1 shows a canonical ordering of G. By the *contour* of G_k we mean its external face $C_k = (w_1 = v_1, w_2, \ldots, w_m = v_2)$. For a given k $(3 \leq k \leq n-1)$, let $w_p, \ldots, w_q$ be the neighbors of $v = v_{k+1}$ in C_k. When we add v to G_k, the edges (w_p, v) and (v, w_q) become contour edges. We call (w_p, v) a *forward edge* and (v, w_q) a *backward edge*. All vertices and edges that disappear from the contour when we add v are said to be *covered* by v. We denote $ind_v(w_i) = i - p + 1$ and call it the *index* of w_i

with respect to v. The *in-degree* $deg^-(v)$ of v is the number of children of v in C_k, that is, $deg^-(v) = q - p + 1$.

A contour vertex w_i $(1 < i < m)$ is called a *valley vertex*, if $w_{i-1} \succ w_i \prec w_{i+1}$; a *peak vertex*, if $w_{i-1} \prec w_i \succ w_{i+1}$. A vertex $v_{k+1} \neq v_1, v_2, v_3$ is called: *forward-oriented*, if $w_p \prec w_{p+1} \prec \ldots \prec w_{q-1} \prec w_q$; *backward-oriented*, if $w_p \succ w_{p+1} \succ \ldots \succ w_{q-1} \succ w_q$; *crossing-valley*, if it covers a valley vertex w_r $(p < r < q)$; *crossing-peak*, if it covers a peak vertex w_r $(p < r < q)$.

We next describe a *generic shift algorithm* in [3] which is the basis of the algorithms in [2, 3, 4, 7, 9] and our algorithm. Given G with canonical ordering π, the algorithm works as follows: We add vertices one at a time according to π. At each step, the contour C_k satisfies certain *contour invariants* that involve restrictions on the slopes of contour edges. When adding a vertex v_{k+1}, we determine its location in the grid and, if necessary, shift some vertices of G_k to the right in order to preserve the contour invariants. We maintain a set $U(v)$ of vertices for each $v \in V$. $U(v)$ contains the vertices located "under" v that need to be shifted whenever v is shifted. The *shift operation* on a contour vertex w_j, denoted by $shift(w_j)$, is achieved by increasing the x-coordinate of each $u \in \cup_{i=j}^{m} U(w_i)$ by 1.

Generic Shift Algorithm:

Initially, place v_1, v_2, v_3 at the points $(0,0), (2,0), (1,1)$, respectively. Let $U(v_i) = \{v_i\}$ $(1 \leq i \leq 3)$.

Suppose G_k $(3 \leq k \leq n-1)$ has been embedded, and we are about to add $v = v_{k+1}$. Let $w_p, \ldots, w_q$ be the children of v in the contour C_k. Define: $U(v_{k+1}) = \{v_{k+1}\} \cup_{i=p+1}^{q-1} U(w_i)$. Apply $shift(w_i)$ to some of $w_1, \ldots, w_m$ (possibly none), so that afterwards there exists at least one point (x', y') satisfying the following conditions, and that placing v at (x', y') preserves the contour invariants.

Definition 3. Generic shift conditions:
(gs1) $x(w_p) \leq x' \leq x(w_q)$;
(gs2) (x', y') is above C_k, i.e. the half line $\{(x', z) | z \geq y'\}$ does not intersect C_k;
(gs3) all vertices $w_p, \ldots, w_q$ are visible from (x', y').

Place v at an arbitrary point (x', y') satisfying these conditions.

Theorem 4. [3] *For all choices of shift operations and vertex coordinates, as long as (gs1), (gs2), (gs3) are satisfied, the Generic Shift Algorithm produces a correct straight line grid embedding.*

3 Drawing Algorithm

Our algorithm crucially depends on the following theorem (proved in [10]).

Theorem 5. *Let G be a triangulated plane graph whose external face $\{v_1, v_2, v_n\}$ is the only non-empty triangle. Then G has a canonical ordering π satisfying the conditions (co1), (co2), (co3), and the following:*

(co4') *Each v_k $(k \leq n-2)$ has at least* **two** *neighbors v_j with $v_j \succ v_k$; v_{n-1} has one neighbor v_n with $v_n \succ v_{n-1}$.*

Moreover, π can be computed in linear time.

Lemma 6. *v_n is the only crossing-peak vertex of G with respect to π.*

Proof. Toward a contradiction, suppose v_{k+1} $(k+1 < n)$ is a crossing-peak vertex with respect to π. Let $w_p, \ldots, w_q$ be the children of v_{k+1} in C_k. Then v_{k+1} covers a peak vertex w_r $(p < r < q)$. So v_{k+1} is the only neighbor of w_r with $w_r \prec v_{k+1}$. This contradicts the condition (co4'). □

Let G be an $(n-1)$ vertex internally triangulated plane graph with no non-empty triangles (obtained from triangulating a 4-connected plane graph). In order to apply Theorem 5 to G, we add a new vertex v_n in the external face F and connect v_n to all vertices on F such that $\{v_1, v_2, v_n\}$ is the external face. The resulting graph G^+ is called the *extended graph* of G. Since G is 4-connected, the only non-empty triangle of G^+ is $\{v_1, v_2, v_n\}$. In the following, we will discuss the embedding of G^+ by using the canonical ordering π of G^+ satisfying Theorem 5, with the understanding that v_n needs not be embedded.

Let n_f and n_b denote the number of forward- and backward-oriented vertices in G^+, among $v_4, \ldots, v_n$. Since v_n is a peak vertex, we have $n_f + n_b \leq n-4$. Without loss of generality, we assume $n_f \leq n_b$. (If not, we vertically "flip" G^+ and swap v_1 and v_2. A forward-oriented vertex in G^+ is a backward-oriented vertex in the flipped graph and vice versa).

Direct each edge $e = (u, v)$ of G^+ from u to v, if $u \prec v$. Denote the resulting directed graph by $\overline{G^+}$. Our algorithm needs a special canonical ordering π_{left} of G^+ obtained as follows. The first three vertices of π_{left} are v_1, v_2, v_3. Suppose the vertex v_k of π_{left} has been defined. Consider a vertex v not in G_k^+. If all incoming neighbors of v are in the contour C_k, v is called a *candidate vertex*. The incoming neighbors of a candidate vertex form a contiguous interval in C_k. Define v_{k+1} of π_{left} to be the candidate vertex whose interval is the leftmost in C_k. This completes the description of π_{left}. Clearly, π_{left} is a canonical ordering of G^+, which is called the *leftmost canonical ordering*. We use $\prec_{left}$ to denote the order defined by π_{left}. (Fig 1 shows the leftmost canonical ordering.)

We need the following concepts introduced in [3]. Each vertex v $(v \neq v_1, v_2)$ of G^+ is classified as either *stable* or *unstable*. With each v, we associate a sequence $DC(v)$ of vertices called its *domino chain*, and a vertex $dom(v)$ called its *dominator*. They are defined as follows. For v_n, $DC(v_n) = (v_n)$, $dom(v_n)$ is undefined, and v_n is stable. Consider $v = v_{k+1}$ $(2 \leq k \leq n-2)$. Let u be the leftmost child of v in C_k. Let z be the vertex that covers the edge (u, v). Then:

Definition 7. Domino chain and dominator:
(dc1) If $ind_z(v) = 2$, then $DC(v) = (v)$, $dom(v) = z$ and v is unstable.
(dc2) If $ind_z(v) \geq 4$, then $DC(v) = (v)$, $dom(v) = z$ and v is stable.
(dc3) If $ind_z(v) = 3$ and $DC(z) = (z_1, \ldots, z_i, z)$, then $DC(v) = (z_1, \ldots, z_i, z, v)$ and $dom(v) = dom(z)$. Also, v is stable if and only if z is stable.

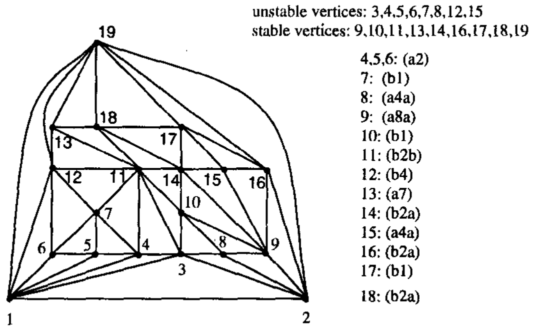

Fig. 1. The leftmost canonical ordering and the grid embedding of G.

As in [3], an unstable vertex of in-degree 2 is called a *room-shift vertex*. The intuition is that a stable vertex can be placed above its leftmost child, while an unstable vertex must be placed at least one x-coordinate to the right of its leftmost child. If v is a room-shift vertex, this can result in putting v directly above its rightmost child w and violating the contour invariants. In this case, we have to shift w to the right in order to "make room" for v.

Example: In Fig 1, $DC(14) = (18,14)$, $dom(14) = 19$. $DC(9) = (10,9)$; $dom(9) = 11$. $DC(3) = (3)$; $dom(3) = 4$.

The slope of an contour edge $e = (w_i, w_{i+1})$ is denoted by $slope(e)$. If $slope(e) = 0$, then e is *horizontal*; If $0 < slope(e) < +\infty$, then e is *upward*; If $slope(e) = +\infty$ or $-\infty$, then e is *vertical*; If $-\infty < slope(e) < 0$, then e is *downward*. We are now ready to describe our algorithm. It is a version of the generic shift algorithm. Our contour invariants are as follows:

Definition 8. Contour invariants:
(ci1) $x(w_1) \leq x(w_2) \leq \ldots \leq x(w_{m-1}) \leq x(w_m)$.
(ci2) Each forward edge (w_{i-1}, w_i) is either horizontal, or upward, or vertical. If w_i is unstable, then (w_{i-1}, w_i) must be horizontal or upward.
(ci3) Each backward edge (w_{i-1}, w_i) is either downward, or horizontal.

Lemma 9. *Let $w_p, \ldots, w_q$ be the children of $v = v_{k+1}$ in the contour C_k. Assume the contour invariants hold. Then $x(w_p) < x(w_{p+1})$. If $deg^-(v) > 2$ and v is unstable, then $x(w_{p+1}) < x(w_{p+2})$.*

Proof. If $e_1 = (w_p, w_{p+1})$ is a backward edge, then e_1 cannot be vertical. If e_1 is a forward edge, then w_{p+1} is unstable and e_1 cannot be vertical. In either case, we have $x(w_p) < x(w_{p+1})$. Suppose $deg^-(v) > 2$ and v is unstable. If $e_2 = (w_{p+1}, w_{p+2})$ is a backward edge, then e_2 cannot be vertical. If e_2 is a forward edge, then since v is unstable and w_{p+2} is the third child of v, w_{p+2} is unstable. Hence e_2 cannot be vertical. In either case, $x(w_{p+1}) < x(w_{p+2})$. □

Suppose we are about to add $v = v_{k+1}$ $(3 \leq k \leq n-2)$. Our algorithm must perform shift operation on some contour vertex (if necessary), and place

v at a point $(x(v), y(v))$ satisfying the generic shift conditions and the contour invariants. This is ensured if $(x(v), y(v))$ satisfies the following:

Definition 10. Placement requirements:
(pr1) $x(v) < x(w_q)$.
(pr2) $x(v) \geq x(w_p)$, if v is stable. $x(v) \geq x(w_p) + 1$, if v is unstable.
(pr3) $y(v) \geq \max\{y(w_p), y(w_q)\}$.
(pr4) $(x(v), y(v))$ is located above C_k, as defined in (gs2) of Definition 3.
(pr5) all vertices $w_p, \ldots, w_q$ are visible from $(x(v), y(v))$.

The placement rules for v depend on: (a) the in-degree of v; (b) v is stable or unstable; (c) v is forward-oriented, backward-oriented, or crossing-valley; (d) the slope of the edges covered by v. Let W and H denote the width and the height of the current grid. First consider the case $deg^-(v) = 2$.

Case (a1): $deg^-(v) = 2$, v is unstable and forward-oriented, (w_p, w_q) is horizontal. v is placed as in Fig 2 (a1): $x(v) = x(w_p) + 1$ and $y(v) = y(w_p) + 1$. If $x(v) = x(w_q)$ perform $shift(w_q)$. W is increased by ≤ 1. H is increased by ≤ 1.

Case (a2): $deg^-(v) = 2$, v is unstable and forward-oriented, (w_p, w_q) is upward. v is placed as in Fig 2 (a2): $x(v) = x(w_p) + 1$ and $y(v) = y(w_q)$. If $x(v) = x(w_q)$ then perform $shift(w_q)$. W is increased by ≤ 1. H is unchanged.

Case (a3): $deg^-(v) = 2$, v is unstable and backward-oriented, the edge (w_p, w_q) is horizontal. v is placed as in Fig 2 (a3): $x(v) = x(w_p) + 1$ and $y(v) = y(w_p) + 1$. If $x(v) = x(w_q)$ then perform $shift(w_q)$. W is increased by at most 1. H is increased by at most 1.

Case (a4a): $deg^-(v) = 2$, v is unstable and backward-oriented, (w_p, w_q) is downward. v is placed as in Fig 2 (a4a): $x(v) = x(w_p) + 1$ and $y(v) = y(w_p)$. If $x(v) = x(w_q)$ perform $shift(w_q)$. W is increased by ≤ 1. H is unchanged.

Case (a4b): Same as (a4a). Fig 2 (a4b) shows an alternative rule: $x(v) = x(w_p) + 1$ and $y(v) = y(w_p) + 1$. If $x(v) = x(w_q)$ then perform $shift(w_q)$. W is increased by at most 1. H is increased by at most 1.

Case (a5): $deg^-(v) = 2$, v is stable and forward-oriented, (w_p, w_q) is horizontal. v is placed as in Fig 2 (a5): $x(v) = x(w_p)$ and $y(v) = y(w_p) + 1$. W is unchanged. H is increased by at most 1.

Case (a6): $deg^-(v) = 2$, v is stable and forward-oriented, the edge (w_p, w_q) is upward. v is placed as in Fig 2 (a6): $x(v) = x(w_p)$ and $y(v) = y(w_q)$. W is unchanged. H is unchanged.

Case (a7): $deg^-(v) = 2$, v is stable and backward-oriented, (w_p, w_q) is horizontal. v is placed as in Fig 2 (a7): $x(v) = x(w_p)$ and $y(v) = y(w_p) + 1$. W is unchanged. H is increased by at most 1.

Case (a8a): $deg^-(v) = 2$, v is stable and backward-oriented, (w_p, w_q) is downward. v is placed as in Fig 2 (a8a): $x(v) = x(w_p) + 1$ and $y(v) = y(w_p)$. If $x(v) = x(w_q)$ then perform $shift(w_q)$. W is increased by ≤ 1. H is unchanged.

Case (a8b): Same as (a8a). Fig 2 (a8b) is an alternative placement of v: $x(v) = x(w_p)$ and $y(v) = y(w_p) + 1$. W is unchanged. H is increased by ≤ 1.

Next consider the case $deg^-(v) \geq 3$. By Lemma 6 and the contour invariants, v's children are embedded as a *V-shaped* polygonal line P. The vertex among w_p and w_q with larger y-coordinate is called the *high-end* vertex of v. Let (w_i, w_{i+1})

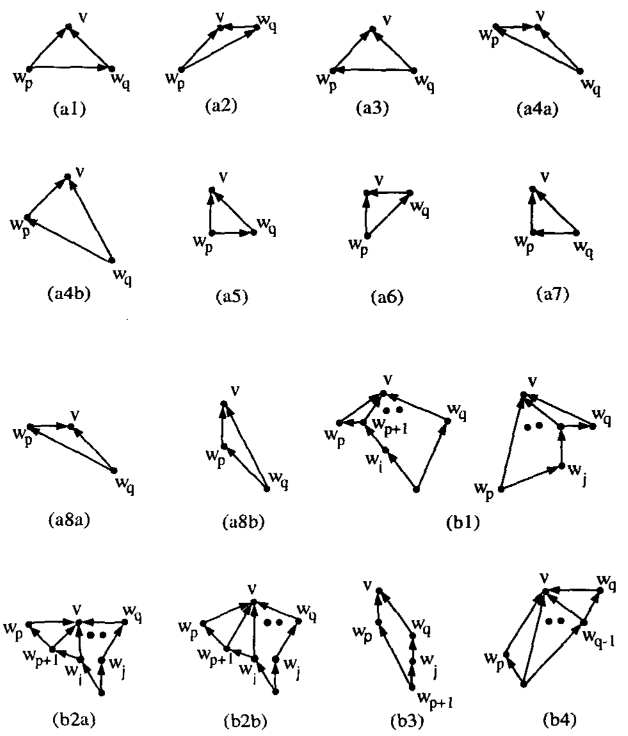

Fig. 2. The placement rules.

be the last downward edge in P. (If no such edge exists, let $w_i = w_p$). Let (w_{j-1}, w_j) be the first upward or vertical edge in P. (If no such edge exists, let $w_j = w_q$). Consider any point $(x(v), y(v))$ such that:

(i) $x(w_i) \leq x(v) < x(w_j)$, if either (w_{j-1}, w_j) or (w_j, w_{j+1}) is vertical; $x(w_i) \leq x(v) \leq x(w_j)$, if neither (w_{j-1}, w_j) nor (w_j, w_{j+1}) is vertical.
(ii) $y(v) \geq \max\{y(w_p), y(w_q)\} + 1$, if the edge adjacent to the high-end vertex of v is horizontal; $y(v) \geq \max\{y(w_p), y(w_q)\}$, if the edge adjacent to the high-end vertex of v is not horizontal.

It is easy to see that such a point satisfies (pr3), (pr4), and (pr5). In the following placement rules, the coordinate $x(v)$, $y(v)$ satisfies the conditions (i), (ii), (pr1) and (pr2). Note that no shift operation is needed for any contour vertex.

Case (b1): $deg^-(v) \geq 3$, the edge covered by v and adjacent to the high-end vertex of v is horizontal. v is placed as in Fig 2 (b1) (two examples are shown): Let $y(v) = \max\{y(w_p), y(w_q)\} + 1$ and determine $x(v)$ as follows. (a) v is stable: Let $x(v) = x(w_i)$. (b) v is unstable: If $w_i = w_p$, let $x(v) = x(w_p) + 1$. If $w_i \neq w_p$, let $x(v) = x(w_i)$. W is unchanged. H is increased by at most 1.

Case (b2a): $deg^-(v) \geq 3$; the edge covered by v and adjacent to the high-end vertex of v is not horizontal; w_p is the high-end vertex of v; and $x(w_j) \geq$

$x(w_p)+2$. v is placed as in Fig 2 (b2a): Let $y(v) = y(w_p)$ and $x(v) = \max\{x(w_p)+1, x(w_i)\}$. W is unchanged. H is unchanged.

Case (b2b): Same as in (b2a). Fig 2 (b2b) shows an alternative rule: Let $y(v) = y(w_p) + 1$, and $x(v) = \max\{x(w_p) + 1, x(w_i)\}$. W is unchanged. H is increased by at most 1.

Case (b3): $deg^-(v) \geq 3$; the edge covered by v and adjacent to the high-end vertex of v is not horizontal; w_p is the high-end vertex of v; and $x(w_j) = x(w_p) + 1$. v is placed as in Fig 2 (b3). In this case, the first edge (w_p, w_{p+1}) covered by v is downward and all other edges covered by v are vertical. So v must be stable. Let $y(v) = y(w_p) + 1$ and $x(v) = x(w_p)$. W is unchanged. H is increased by at most 1.

Case (b4): $deg^-(v) \geq 3$; the edge covered by v and adjacent to the high-end vertex of v is not horizontal; w_q is the only high-end vertex of v. v is placed as in Fig 2 (b4): Let $y(v) = y(w_q)$. If v is stable, let $x(v) = x(w_i)$. If v is unstable, let $x(v) = \max\{x(w_p) + 1, x(w_i)\}$. W is unchanged. H is unchanged.

In the cases (a4), (a8), and (b2), two alternatives are given. They are chosen as follows: Suppose that $v = v_{k+1}$ $(k + 1 < n - 1)$ satisfies the conditions of the rule (a4) (or (a8) or (b2), resp.) Let u be the vertex that covers the edge (w_p, v). Let z_1, z_2, z_3 be the first, second, and third child of u. If the following conditions hold, we must place v by using the rule (a4b), (or (a8b) or (b2b), resp.)

Definition 11. Avoid-horizontal-forward-edge conditions:
(1) $deg^-(u) \geq 3$;
(2) (w_p, v) is the last edge covered by u;
(3) either of the following two conditions hold: (3a) the edge (z_1, z_2) is upward; or (3b) $z_3 \neq v$, the edge (z_2, z_3) is upward, and $y(z_1) \leq y(w_p)$.

Remark 1: The rules (a4a), (a8a) and (b2a) are the only rules that create horizontal forward edges. By using the rules (a4b), (a8b) and (b2b), this can be avoided. More precisely, if v is placed by using the rules (a4a), (a8a) or (b2a), the edge (w_p, v) is horizontal and v is not a peak point (see Fig 3(1)). If v is placed by using the rules (a4b), (a8b) or (b2b), the edge (w_p, v) is either upward or vertical, and v is a peak point (Fig 3 (2)).

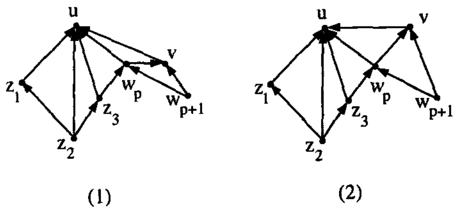

Fig. 3. The avoid-horizoantal-forward-edge conditions.

Remark 2: By the definition of π_{left}, z_1, z_2, z_3 are embedded before v. So the avoid-horizontal-forward-edge conditions can be checked when v is embedded.

For a vertex v satisfying the conditions of the rule (a4) (or (b2), respectively), if the avoid-horizontal-forward-edge conditions do not hold, then we place v by using the rule (a4a) (or (b2a), respectively). For a vertex v satisfying the conditions of the rule (a8), if the avoid-horizontal-forward-edge conditions do not hold, v is called a *free-lance* vertex. A free-lance vertex can be placed by using either the rule (a8a) or (a8b). We use this freedom to adjust the height and the width of the embedding as follows. Let n_r be the number of room-shift vertices. Let $d = (n/2 - 1) - n_r$. We place the first (at most) d free-lance vertices by using the rule (a8a). All other (if any) free-lance vertices are placed by using the rule (a8b). This completes the description of our algorithm.

Example: The graph G in Fig 1 is embedded by using our algorithm. (Recall $v_n = 19$ is not embedded). The rule used for each vertex is as indicated. The free-lance vertex 9 is placed by using the rule (a8a). The vertex 11 is placed by using the rule (b2b) since it satisfies the avoid-horizontal-forward-edge conditions.

4 Bounding the Grid Size

Let W and H be the width and the height of the final grid. If v is placed by using the rule (ai) ($i = 1, 2, 3, 4a, 4b, 5, 6, 7, 8a, 8b$) or the rule (b$i$) ($i = 1, 2a, 2b, 3, 4$), we call v an (ai) or a (bi) vertex. Let a_i be the number of (ai) vertices and b_i be the number of (bi) vertices. A vertex that is the first one reaching a new y-coordinate is called a *height-increasing* vertex. A height-increasing vertex must be either an (ai) vertex (for $i = 1, 3, 4b, 5, 7, 8b$), or a (bi) vertex (for $i = 1, 2b, 3$). A height-increasing vertex of type (ai) or (bi) is called an (a'i) or a (b'i) vertex. Let a'_i and b'_i be the number of (a'i) and (b'i) vertices, respectively. We have:

$$H = 1 + a'_1 + a'_3 + a'_{4b} + a'_5 + a'_7 + a'_{8b} + b'_1 + b'_{2b} + b'_3 \tag{1}$$

If the placement of v increases W, v is a *width-increasing* vertex. A *width-increasing* vertex must be an (ai) vertex for $i = 1, 2, 3, 4a, 4b, 8a$. Let a''_i ($i = 1, 2, 3, 4a, 4b, 8a$) be the number of (ai) vertices that are **not** width-increasing:

$$W = 2 + (a_1 + a_2 + a_3 + a_{4a} + a_{4b} + a_{8a}) - (a''_1 + a''_2 + a''_3 + a''_{4a} + a''_{4b} + a''_{8a}) \tag{2}$$

Since v_1, v_2, v_3 are neither (ai) nor (bi) vertices and v_n is not embedded, we have:

$$a_1 + a_2 + a_3 + a_{4a} + a_{4b} + a_5 + a_6 + a_7 + a_{8a} + a_{8b} + b_1 + b_{2a} + b_{2b} + b_3 + b_4 = n - 4 \tag{3}$$

Bound on $H + W$: From equations (1), (2), and (3), we have:

$$\begin{aligned} W + H &= 3 + (a_1 + a_2 + a_3 + a_{4a} + a_{4b} + a'_5 + a_6 + a'_7 + a_{8a} + a'_{8b} + b'_1 + b_{2a} \\ &\quad + b'_{2b} + b'_3 + b_4) + (a'_1 + a'_3 + a'_{4b}) - (a_6 + b_{2a} + b_4) - \\ &\quad (a''_1 + a''_2 + a''_3 + a''_{4a} + a''_{4b} + a''_{8a}) \\ &\leq (n - 1) + (a'_1 + a'_3 + a'_{4b}) - (a_6 + b_{2a} + b_4) - \\ &\quad (a''_1 + a''_2 + a''_3 + a''_{4a} + a''_{4b} + a''_{8a}) \end{aligned} \tag{4}$$

Let D denote the set of (a1), (a3), (a4b) vertices; D' the set of (a'1), (a'3), (a'4b) vertices; and J the set of (a6), (b2a), (b4) vertices. Thus $|D'| = a'_1 + a'_3 + a'_{4b}$

and $|J| = a_6 + b_{2a} + b_4$. A vertex in D' may increase both W and H by 1. A vertex in J increases neither W nor H. We will show $|D'| \leq |J|$. Define:

$K = \{v | v$ is unstable and (w_p, v) is upward, where w_p is the leftmost child of $v\}$

Note that $D' \subseteq D \subseteq K$. For each $v \in K$, we define a sequence of vertices $S(v) = (x_0 = v, x_1, \ldots, x_k, u)$ (possibly $k = 0$) such that:
(1) For each i $(1 \leq i \leq k)$, x_i is an (a2) vertex and covers the edge (w_p, x_{i-1}).
(2) u is either an (a6), (b4), (b2a), or (b2b) vertex and covers the edge (w_p, x_k).

$S(v)$ is defined as follows. Start with $S = (x_0 = v)$. Suppose that x_i has been defined. Let u be the vertex that covers the edge (w_p, x_i), where w_p is the leftmost child of x_i. There are three cases:

Case 1: $deg^-(u) = 2$ and u is unstable. Define $x_{i+1} = u$, and continue. Note that (w_p, x_i) is upward. By our rules, x_{i+1} must be an (a2) vertex.

Case 2: $deg^-(u) = 2$ and u is stable. Let $k = i$ and u be the last vertex of $S(v)$ and we are done. (Since (w_p, x_k) is upward, u must be an (a6) vertex).

Case 3: $deg^-(u) \geq 3$. Let $k = i$ and u be the last vertex of $S(v)$ and we are done. Since u covers the edge (w_p, x_k) and x_k is unstable, there are two subcases:

Case 3A: $ind_u(x_k) = 2$. Since the edge (w_p, x_k) is upward, u is forward-oriented (Fig 4(1)). By the avoid-horizontal-forward-edge conditions, the last edge covered by u is either upward or vertical. So u is a (b4) vertex.

Case 3B: $ind_u(x_k) = 3$ and u is unstable. Let $e_1 = (t_1, w_p)$ and $e_2 = (t_2, t_3)$ be the first and the last edge covered by u (see Fig 4(2)).

(i) Suppose $deg^-(u) > 3$ and $y(t_1) \leq y(t_2)$. By the avoid-horizontal-forward-edge conditions, e_2 is upward or vertical and t_3 is the only high-end vertex of u. So u is placed as a (b4) vertex.

(ii) Suppose $deg^-(u) > 3$ and $y(t_1) > y(t_2)$. If e_2 is horizontal, then t_1 is the only high-end vertex of u and e_1 is not horizontal. If e_2 is upward, then neither e_1 nor e_2 is horizontal. So u is either a (b2a), or (b2b), or (b4) vertex.

(iii) Suppose $deg^-(u) = 3$. Then $(w_p, x_k) = (t_2, t_3)$. Depending on which of t_1 and t_3 is the high-end vertex of u, u is either a (b2a), (b2b) or a (b4) vertex.

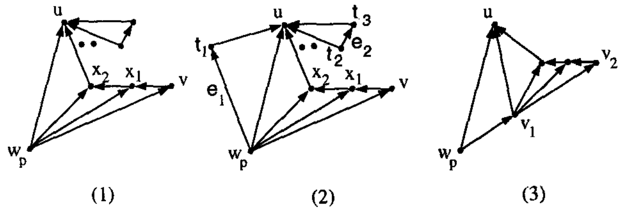

Fig. 4. The last vertex u in the sequence $S(v)$.

Note that in $S(v) = \{x_0 = v, x_1, \ldots, x_k, u\}$, the vertices $x_1, \ldots, x_k$ (if any) are uniquely determined by v. On the other hand, a vertex u can be the last vertex of $S(v_1)$ and $S(v_2)$ for two distinct vertices $v_1, v_2 \in K$, where u satisfies Case 3A for $S(v_1)$ and Case 3B for $S(v_2)$. (See Fig 4(3)).

We construct a directed forest F such that the following hold: (a) The node set of F is a subset of the vertices of G^+; (b) The set of leaf nodes in F is the set D; (c) Each non-root internal node of F is a (b4) vertex, and has exactly two children; (d) Each root node of F is either a (b2b) vertex or in J, and has exactly one child.

The forest F is constructed as follows. We keep a set $Q \subseteq K$. For each $v \in Q$, we find the sequence $S(v)$ and define $parent(v)$ to be the last vertex u of $S(v)$. When a vertex v is put in Q, $parent(v)$ is identified. At any moment, for each vertex $v \in Q$, $parent(v)$ is the root of a tree. Initially, let $Q = D$.

By the remark above, at most two vertices v_1, v_2 in Q can have the same parent u. In this case, u must be unstable and the edge (w_p, u) (where w_p is the leftmost child of u) is upward. (See Fig 4 (3)). So u is a (b4) vertex and is also in K. In this case, we remove v_1, v_2 from Q and put u into Q.

Repeat this until all vertices in Q have distinct parents. Now the parents of the vertices in Q are the roots of the trees of the forest F to be constructed. It is easy to check that F satisfies all above conditions.

Consider a tree T in F. Let $Leaf(T)$ and $Int(T)$ be the number of leaf nodes and internal nodes in T, respectively. Clearly, $Leaf(T) = Int(T)$. Let $D(T)$, $D'(T)$, and $J(T)$ be the number of nodes in T that are in D, D', and J, respectively. We will show $D'(T) \leq J(T)$.

If the root of T is in J, then $D'(T) \leq D(T) = Leaf(T) = Int(T) = J(T)$. Suppose the root of T is a (b2b) vertex. Let v be the leaf node in T that has the smallest y-coordinate among all leaf nodes in T. Then v is not a height-increasing vertex. Thus v is in D but not in D'. Hence, $D'(T) \leq D(T) - 1 = Leaf(T) - 1 = Int(T) - 1 = J(T)$.

Since each vertex in D' corresponds to a distinct leaf node in F and $D'(T) \leq J(T)$ holds for every tree T in F, we have: $|D|' \leq |J|$. By (4), this gives:

$$W + H \leq (n-1) - (a_1'' + a_2'' + a_3'' + a_{4a}'' + a_{4b}'' + a_{8a}'') \leq (n-1) \qquad (5)$$

Bound on the width W**:** We first bound n_r (the number of room-shift vertices). Each room-shift vertex v is associated with a vertex $dom(v)$. It was shown in [3] that, for two distinct room-shift vertices v_1 and v_2, $dom(v_1) \neq dom(v_2)$. By the definition of the dominator, only the forward-oriented vertices and v_n can be dominators of room-shift vertices. By our assumption on canonical ordering, we have $n_f \leq n_b$. Hence:

$$n_r = a_1 + a_2 + a_3 + a_{4a} + a_{4b} \leq n_f + 1 \leq (n-4)/2 + 1 = n/2 - 1$$

Thus $d = (n/2 - 1) - n_r \geq 0$. Our algorithm places at most d free-lance vertices by using the rule (a8a). So $a_{8a} \leq d$. From equation (2), we have:

$$W \leq 2 + (a_1 + a_2 + a_3 + a_{4a} + a_{4b}) + a_{8a} \leq 2 + n_r + (n/2 - 1) - n_r = n/2 + 1 \quad (6)$$

Bound on the heigh H**:** First suppose the number of free-lance vertices is at least $d = (n/2 - 1) - n_r$. Then our algorithm places $a_{8a} = d$ of them by using the rule (a8a). So $a_1 + a_2 + a_3 + a_{4a} + a_{4b} + a_{8a} = n_r + d = n/2 - 1$. By (2) and (5), we have:

$$H \leq (n-1) - (a_1'' + a_2'' + a_3'' + a_{4a}'' + a_{4b}'' + a_{8a}'') - W = (n-1) - (2 + a_1 + a_2 + a_3 + a_{4a} + a_{4b} + a8a) = (n-1) - (n/2+1) < 2(n-2)/3 \qquad (7)$$

Next suppose the number of free-lance vertices is less than d. Then no free-lance vertices are placed by using the rule (a8b). So all (a8b) vertices satisfy the avoid-horizontal-forward-edge conditions.

Let A denote the set of (a'1), (a'3), (a'5), (a'7), and (b'1) vertices; B the set of (a'4b), (a'8b), (b'2b) vertices; and C the set of (b'3) vertices. From equation (1), we have: $H = 1 + |A| + |B| + |C|$. For each $v \in A \cup B \cup C$, we define a vertex $mate(v)$ as follows. Consider a vertex $v \in A$. v covers a horizontal edge (x, y) with y-coordinate $y(v) - 1$. (See Fig 5(1)). Define $mate(v)$ to be the vertex among x and y that is not a height-increasing vertex.

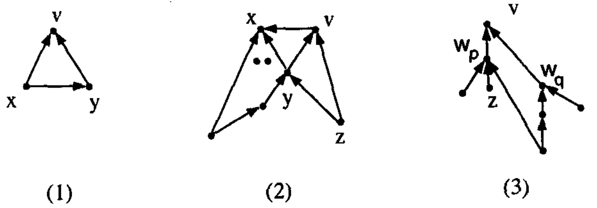

Fig. 5. The definition of the mate vertices.

Consider a vertex $v \in B$. Let y be the first child of v. Let x be the vertex that covers the edge (y, v) (Fig 5(2)). v is a peak point and (x, v) is horizontal. So x is not height-increasing. Define $mate(v) = x$.

Consider a vertex $v \in C$. Let w_p and w_q be the leftmost and the rightmost child of v, respectively (Fig 5(3)). By the rule (b3), we have $y(w_p) \geq y(w_q)$. By the definition of π_{left}, we have $w_p \prec_{left} w_q$. Thus there is a vertex z such that $z \prec_{left} w_p \prec_{left} w_q$ and $y(z) = y(w_q)$. So w_q is not height-increasing. Define $mate(v) = w_q$.

Define: $Mate = \{mate(v) \mid v \in A \cup B \cup C\}$. From the definition, it is easy to check that: (i) If $x \in Mate$, then x is not in $A \cup B \cup C$. (ii) If $x = mate(v)$ for a vertex $v \in C$, then x cannot be $mate(v')$ for any $v' \neq v$. (iii) A vertex x can be the mate vertex for at most two vertices: one $v \in A$ and another $v' \in B$. Hence each vertex in $Mate$ corresponds to at most two vertices in $A \cup B \cup C$. Thus:

$$|A| + |B| + |C| \leq 2|Mate|$$

The vertices v_1, v_2, v_3, v_n are neither in $A \cup B \cup C$ nor in $Mate$. Thus: $|A| + |B| + |C| + |Mate| \leq (n-4)$. So $|A| + |B| + |C| \leq 2(n-4)/3$. From (1), this gives:

$$H = 1 + (|A| + |B| + |C|) \leq 1 + 2(n-4)/3 < 2(n-2)/3 \qquad (8)$$

Recall that G^+ has n vertices, and the original graph G has $n-1$ vertices. So the bounds in (5), (6), (7), (8) imply the bounds on size in Theorem 1.

Implementation of the algorithm: The graph G^+ can be constructed from G in linear time. The canonical ordering π can be computed in $O(n)$ time [10]. The leftmost canonical ordering π_{left} can be computed from π in linear time by using the method in [11]. After π_{left} is known, the algorithm can be implemented by using the method in [3]. So the algorithm takes linear time. This completes the proof of Theorem 1.

References

1. 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.), 1982, pp.153–173.
2. M. Chrobak and G. Kant, Convex grid drawings of 3-connected planar graphs, Tech. Rep. RUU-CS-93-45, Dept. of Comp. Sci. Utrecht University, 1993.
3. M. Chrobak and S. Nakano, Minimum-width grid drawings of planar graphs, in Proc. Workshop on Graph Drawing'94, Princeton, NJ, Oct. 1994.
4. M. Chrobak and T. Payne, A linear time algorithm for drawing planar graphs on a grid, TR UCR-CS-89-1, Dept. of Math. and Comp. Sci., UC at Riverside, 1989.
5. G. Di Battista, P. Eades, R. Tamassia, and I. G. Tollis, Algorithms for drawing graphs: an annotated bibliography, Comput. Goem. Theory Appl. Vol 4, 1994, pp. 235-282.
6. I. Fáry, On straight line representation of planar graphs, Acta. Sci. Math. Szeged, 11, 1948, pp. 229–233.
7. H. de Fraysseix, J. Pach and R. Pollack, How to draw a planar graph on a grid, *Combinatorica* 10, 1990, pp. 41–51.
8. M. Fürer, X. He, M. Y. Kao, and B. Raghavachari, $O(n \log\log n)$-work parallel algorithms for straight line grid embeddings of planar graphs, SIAM J. Disc. Math 7(4), 1994, pp. 632-647.
9. G. Kant, Drawing planar graphs using the *lmc*-ordering, in *Proc. 33th Ann. IEEE Symp. on Found. of Comp. Science*, Pittsburgh, 1992, pp. 101-110.
10. G. Kant and X. He, Two algorithms for finding rectangular duals of planar graphs, in Proc. 19th on Graph-Theoretic Concepts in CS, 1993, LNCS 790, pp. 396-410.
11. F. Preparata and R. Tamassia, Fully dynamic techniques for point location and transitive closure in planar structures, in Proc. 29th FOCS, 1988, pp. 558-567.
12. P. Rosenstiehl and R. Tarjan, Rectilinear planar layouts and bipolar orientations of planar graphs, Discrete & Computational Geometry 1, 1986, pp. 343–353.
13. W. Schnyder, Embedding planar graphs on the grid, Abs. AMS 9, 1988, p. 268.
14. W. Schnyder, Planar graphs and poset dimension, Orders 5, 1989, pp. 323–343.
15. W. Schnyder, Embedding planar graphs on the grid, in Proc. of the 1st Annual ACM-SIAM Symp. on Discrete Algorithms, 1990, pp. 138–147.
16. W. Schneider and W. Trotter, Convex drawings of planar graphs, Abstracts of AMS 13 (5), 1992.
17. S. K. Stein, Convex maps, in Proc Amer Math Soc, Vol. 2, 1951, pp. 464–466.
18. W. T. Tutte, How to draw a graph, Proc. London Math. Soc. 13, 1963, pp. 743–768.
19. K. Wagner, Bemerkungen zum Vierfarben problem, Jahresbericht Deutsch Math 46, 1936, pp. 26–32.