Grid Embedding of Internally Triangulated Plane Graphs without Non-empty Triangles *

Xin He
Department of Computer Science
State University of New York at Buffalo
Buffalo, NY 14260
xinhe@cs.buffalo.edu

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 an internally triangulated plane graph G has no non-empty triangles (a non-empty triangle is a triangle of G containing some vertices in its interior), 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

Let $G = (V, E)$ be a graph with n vertices. We always assume $n \geq 3$ in this paper. G is *planar* if it can be drawn on the plane such that the vertices are located at distinct points, and the edges are represented by nonintersecting curves joining their endpoints. A *plane graph* is a planar graph with a fixed plane embedding. A *straight line grid embedding* of a 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 and have applications in CAD/CAM and Computer Graphics [6].

Wagner [24], Fáry [8], and Stein [22] independently showed that every planar graph has a straight line embedding. Since then, many embedding algorithms have been reported (e.g. [23, 2, 16]). The earlier algorithms all suffer two serious drawbacks that render them useless in practice, as noted by de Fraysseix, Pach, and Pollack [9]. First, they require high-precision real arithmetic relative to the size of the graph, 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

*The work was partially supported by National Science Foundation, grant CCR-9205982.

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 [17] posed the problem of computing a straight line embedding on a grid of polynomial size. Schnyder [19] 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 [9, 10]. The running time of their algorithm was improved by Chrobak and Payne to $O(n)$ [5]. Schnyder further proved the existence of an embedding on an $(n-2) \times (n-2)$ grid [18] and gave an $O(n)$ time algorithm to compute such an embedding [20]. Schnyder's algorithm can be implemented in parallel in $O(\log n \log\log n)$ time with $n/\log n$ processors on a parallel random access machine [11]. It was shown in [12] 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 polygons. The grid size of such embedding is reduced to $(n-2) \times (n-2)$ in [3, 13, 21].

It has been observed that there exists a plane graph G, as shown in Fig 1, such that the straight line grid embedding of G requires a grid of size at least $(2n/3-1) \times (2n/3-1)$ [9]. Further, it was shown 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 [4]. It has been conjectured that every planar graph can be embedded on a $2n/3 \times 2n/3$ grid. While this conjecture remains open, some progress has been made recently: 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 [4].

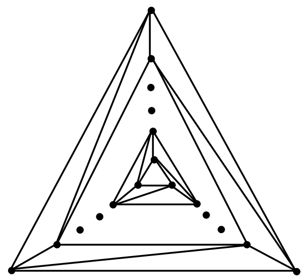

Figure 1: A graph that requires a $(2n/3-1) \times (2n/3-1)$ grid.

For the straight line grid embedding problem, we can assume, without loss of generality, that all internal faces of G are triangles. (If not, we can add dummy edges so that all internal faces of G are triangles. After an embedding is obtained, the dummy edges can be removed). If all internal faces of G are triangles, it is called an *internally triangulated plane* graph. If the external face of G is also a triangle, then G is called a *triangulated plane* graph. From now on, we only consider internally triangulated plane graphs.

Given such a graph G, a *non-empty triangle* of G is a triangle of G containing some vertices in its interior. Note that the graph G shown in Fig 1 consists of many levels of nested non-empty triangles. In this paper we show that if G has no non-empty triangles, then the above mentioned

$(2n/3-1) \times (2n/3-1)$ lower bound on grid size does not hold. We will prove:

Theorem 1.1 *Every n vertex internally triangulated plane graph without non-empty triangles 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 particular, any 4-connected plane graph G with at least four vertices on its external face can be internally triangulated so that the resulting graph G' satisfies the conditions in Theorem 1.1. (If G' has a non-empty triangle $\{x, y, z\}$, then $\{x, y, z\}$ would be a separating triple of G. This contradicts the 4-connectivity of G). So every such graph has a grid embedding as stated in Theorem 1.1.

The present paper is organized as follows. In Section 2, we review some definitions and briefly describe a *generic shift algorithm* in [4], which is the basis of our algorithm. In Section 3, we present our algorithm. In section 4, we prove Theorem 1.1. Section 5 concludes the paper.

2 Preliminaries

Most graph-theoretic definitions used here are standard (e.g. [1, 7]). The embedding algorithms in [9, 10, 5, 4, 12, 13] are all based on the following *canonical ordering* concept introduced in [9].

Definition 2.1 *Let $G = (V, E)$ be a triangulated plane graph with n vertices. Let $\pi = v_1, v_2, \ldots, v_n$ be 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 conditions hold for each $k = 3, \ldots, n$:*

(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 [9, 10]. We will use the symbol $\prec$ to denote the linear order of the canonical ordering. Namely, if $i < j$ then we write $v_i \prec v_j$. Note that the condition (co4) is implied by the condition (co3). For the sake of clarity, we make (co4) an explicit condition. Fig 2 shows a canonical ordering of a triangulated plane graph G.

By the *contour* of G_k we mean its external face C_k written as: $C_k = (w_1 = v_1, w_2, \ldots, w_m = v_2)$. 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 leftmost edge (w_p, v) and the rightmost edge (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 call v the *cover vertex* of w_j for $p < j < q$. We call w_p the *first child* of v; w_{p+1} the *second child* of v ... and w_q the *last child* of v. We denote $ind_v(w_i) = i - p + 1$

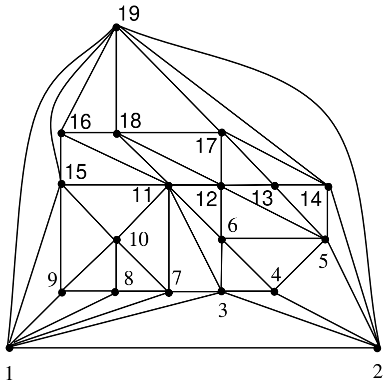

Figure 2: A triangulated plane graph G and its canonical ordering.

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, $deg^-(v) = q - p + 1$. If $n \geq 4$, clearly $deg^-(v_k) \geq 2$ for $4 \leq k \leq n-1$ and $deg^-(v_n) \geq 3$.

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 $(p < r < q)$.

In particular, a vertex of in-degree 2 is forward-oriented (backward-oriented, respectively) if and only if it covers a forward (backward, respectively) edge. A vertex of in-degree ≥ 4 can be both a crossing-valley and a crossing-peak vertex.

We next briefly describe a *generic shift algorithm* in [4] which is the basis of the embedding algorithms in [3, 4, 5, 9, 10, 12, 13] and our algorithm.

Given a triangulated plane graph $G = (V, E)$ with canonical ordering π, the genetic shift 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 have to determine which vertices of G_k must be shifted to the right. This is done by maintaining 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 vertex $u \in \cup_{i=j}^{m} U(w_i)$ 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 that G_k ($3 \le k \le n-1$) has already been embedded, and we are about to add $v = v_{k+1}$. Let $C_k = (w_1, \ldots, w_m)$ be the contour of G_k and $w_p, \ldots, w_q$ be the children of v in 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 grid point (x', y') satisfying the following conditions, and that if v is placed at (x', y') then the contour invariants are preserved.

Definition 2.2 Generic shift conditions:

(gs1) *$x(w_p) \le x' \le x(w_q)$;*

(gs2) *(x', y') is located above C_k in the sense: the half line $\{(x', z) \mid z \ge y'\}$ does not intersect C_k;*

(gs3) *all vertices $w_p, \ldots, w_q$ are visible from (x', y'). (Namely, the edges from v to all $w_p, \ldots, w_q$ do not intersect each other, nor the edges in C_k).*

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

End Generic Shift.

The following theorem was implicit in [9, 10, 5] and explicitly stated in [4].

Theorem 2.1 *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.*

To use the generic shift algorithm, one needs to specify the contour invariants and describe how to place the vertices so that both the contour invariants and the generic shift conditions are satisfied.

3 Drawing Algorithm

Our algorithm crucially depends on the following theorem which was proved in [14].

Theorem 3.1 *Let $G = (V, E)$ be a triangulated plane graph whose external face $\{v_1, v_2, v_n\}$ is the only non-empty triangle of G. Then G has a canonical ordering π satisfying the conditions (co1), (co2), (co3), and the following condition:*

(co4') *If $k \le n-2$, then v_k has at least* **two** *neighbors v_j with $v_j \succ v_k$. The vertex v_{n-1} has one neighbor v_n with $v_n \succ v_{n-1}$.*

Moreover, π can be computed in linear time.

The canonical ordering π of G in Theorem 3.1 has the following property.

Lemma 3.2 *The vertex 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 $G = (V, E)$ be an $(n-1)$ vertex internally triangulated plane graph with no non-empty triangles. In this section we present our algorithm for embedding G.

In order to apply Theorem 3.1 to G, we need to modify G so that its external face is the only non-empty triangle. Let $x_1, x_2, \ldots, x_m$ be the external face F of G. An edge $e = (x_i, x_j)$ ($i < j$) is a *chord*, if $1 < j - i < n - 1$. e is a *minimal chord* if the cycle $\{x_i, x_{i+1}, \ldots, x_j\}$ has no chords. Let $e = (x_i, x_j)$ be a minimal chord. It is easy to show that we can add either the edge (x_{i-1}, x_{i+1}) or the edge (x_{j-1}, x_{j+1}) into F without introducing non-empty triangles. (The only exception to this claim is when G is a 4-cycle with a chord. Since G can be easily embedded in a 1×1 grid, we don't consider this case). Repeat this process until all chords on the external face are eliminated. Now 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. Clearly, 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 the conditions in Theorem 3.1, with the understanding that the vertex v_n needs not be embedded and all added edges will be removed after the embedding is obtained. Note that the number of vertices in the original graph G is $n-1$.

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. Clearly, π is a canonical ordering of the flipped graph and Theorem 3.1 still holds).

Direct the edges of G^+ as follows: Direct an edge $e = (u, v)$ from u to v, if $u \prec v$. We say the resulting directed acyclic graph $\overline{G^+}$ is *derived* from π. It is easy to see that the canonical ordering π is a topological ordering of $\overline{G^+}$, and any topological ordering of $\overline{G^+}$ is a canonical ordering of G^+.

When the generic shift algorithm is performed on G^+, if there is a directed path P from a vertex u to a vertex v in $\overline{G^+}$, then u must be drawn before v (and vice versa). If no such P exists, however, the order in which u and v are drawn is irrelevant.

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^+. We say u is an *incoming neighbor* of v, if there is an edge directed from u to v in $\overline{G^+}$. If all incoming neighbors of v are in the contour C_k of G_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 topological ordering of $\overline{G^+}$ and hence a canonical ordering of G^+. We will use the symbol $\prec_{left}$ to denote the order defined by π_{left}. We call π_{left} the *leftmost canonical ordering* of G^+

derived from π. (Note that π_{left} is different from the leftmost canonical ordering defined in [12, 13]). Our algorithm will use π_{left}, which can be computed in linear time from π by using the method in [15]. According to π_{left}, if there are more than one candidate vertex for G^+_k, the leftmost one is added first. Fig 3 shows the leftmost canonical ordering of the graph shown in Fig 2.

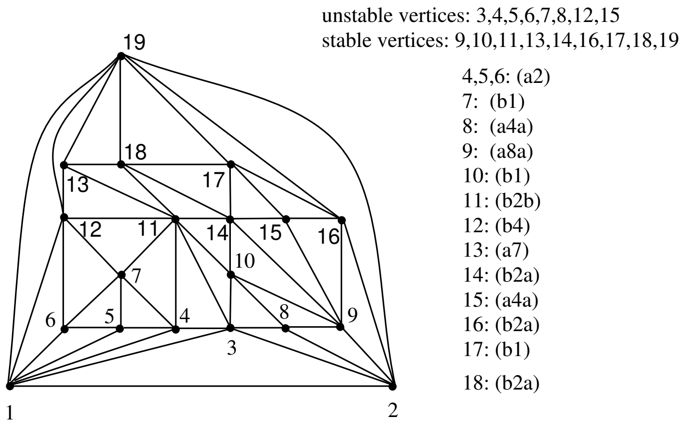

Figure 3: The leftmost canonical ordering and the grid embedding of G.

We need the following concepts introduced in [4]. Each vertex v ($v \neq v_1, v_2$) of G^+ is classified as either *stable* or *unstable*. With each such vertex 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, we define $DC(v_n) = (v_n)$, $dom(v_n)$ 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, that is $ind_v(u) = 1$. Let z be the vertex that covers the edge (u, v). Such z must exist because $v \neq v_n$. Then:

Definition 3.1 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.

As in [4], 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, saving one x-coordinate, 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 3, the unstable and the stable vertices of G are shown. $DC(14) = (18, 14)$, $dom(14) = 19$. $DC(9) = (10, 9)$; $dom(9) = 11$. $DC(3) = (3)$; $dom(3) = 4$.

Let $C_k = (w_1, \ldots, w_m)$ be the contour of G_k^+. The location of a vertex w_i in the grid will be referred to as the *point* w_i. A contour point w_i is a *peak point* if $y(w_{i-1}) < y(w_i)$ and $y(w_i) > y(w_{i+1})$. w_i is a *valley point* if $y(w_{i-1}) > y(w_i)$ and $y(w_i) < y(w_{i+1})$. The slope of a contour edge $e = (w_i, w_{i+1})$ is denoted by $slope(e)$. We classify e as follows:

- 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 3.2 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 (thus* $x(w_i) \geq x(w_{i-1}) + 1$*).*

(ci3) *Each backward edge* (w_{i-1}, w_i) *is either downward, or horizontal (thus* $x(w_i) \geq x(w_{i-1}) + 1$*).*

The following lemma was proved in [4]. We include its proof here for completeness.

Lemma 3.3 *Let* $w_p, \ldots, w_q$ *be the children of the vertex* $v = v_{k+1}$ *in the contour* C_k*. Assume the contour invariants are satisfied. 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, we have $x(w_{p+1}) < x(w_{p+2})$. □

Suppose we are about to add $v = v_{k+1}$ $(3 \leq k \leq n-2)$. (Recall that the vertex v_n is not embedded). Let $C_k = (w_1, \ldots, w_m)$ be the contour of G_k^+ and $w_p, \ldots, w_q$ be the children of v in C_k. 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 such that the contour invariants are preserved after v is placed. This is ensured, as one can easily check, if $(x(v), y(v))$ satisfies the following:

Definition 3.3 Placement requirements:

(pr1) $x(v) < x(w_q)$. *(This is required because the edge* (v, w_q) *cannot be vertical).*

(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 the condition (gs2) in Definition 2.2.*

(pr5) *all vertices* $w_p, \ldots, w_q$ *are visible from* $(x(v), y(v))$.

The placement rules for v depend on the following factors:

- The in-degree of v.
- v is stable or unstable.
- v is forward-oriented, or backward-oriented, or crossing-valley.
- The edges covered by v are horizontal, vertical, upward, or downward.

Let W and H denote the width and the height, respectively, of the current grid. In the following placement rules, we will indicate how the values of W and H change after v is placed. First consider the placement rules for $v = v_{k+1}$ when $deg^-(v) = 2$. Let w_p and w_q be the leftmost and the rightmost child of v. Note that the edge (w_p, w_q) cannot be vertical by Lemma 3.3.

Case (a1): $deg^-(v) = 2$, v is unstable and forward-oriented, the edge (w_p, w_q) is horizontal. v is placed as in Fig 4 (a1): $x(v) = x(w_p) + 1$ 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 (a2): $deg^-(v) = 2$, v is unstable and forward-oriented, the edge (w_p, w_q) is upward. v is placed as in Fig 4 (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 at most 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 4 (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, the edge (w_p, w_q) is downward. v is placed as in Fig 4 (a4a): $x(v) = x(w_p) + 1$ and $y(v) = y(w_p)$. If $x(v) = x(w_q)$ then perform $shift(w_q)$. W is increased by at most 1. H is unchanged.

Case (a4b): Same as case (a4a). Fig 4 (a4b) shows an alternative placement of v: $x(v) = x(w_p) + 1$ 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, the edge (w_p, w_q) is horizontal. v is placed as in Fig 4 (a5): $x(v) = x(w_p)$ and $y(v) = y(w_p) + 1$. W is unchanged. H is increased by at most 1.

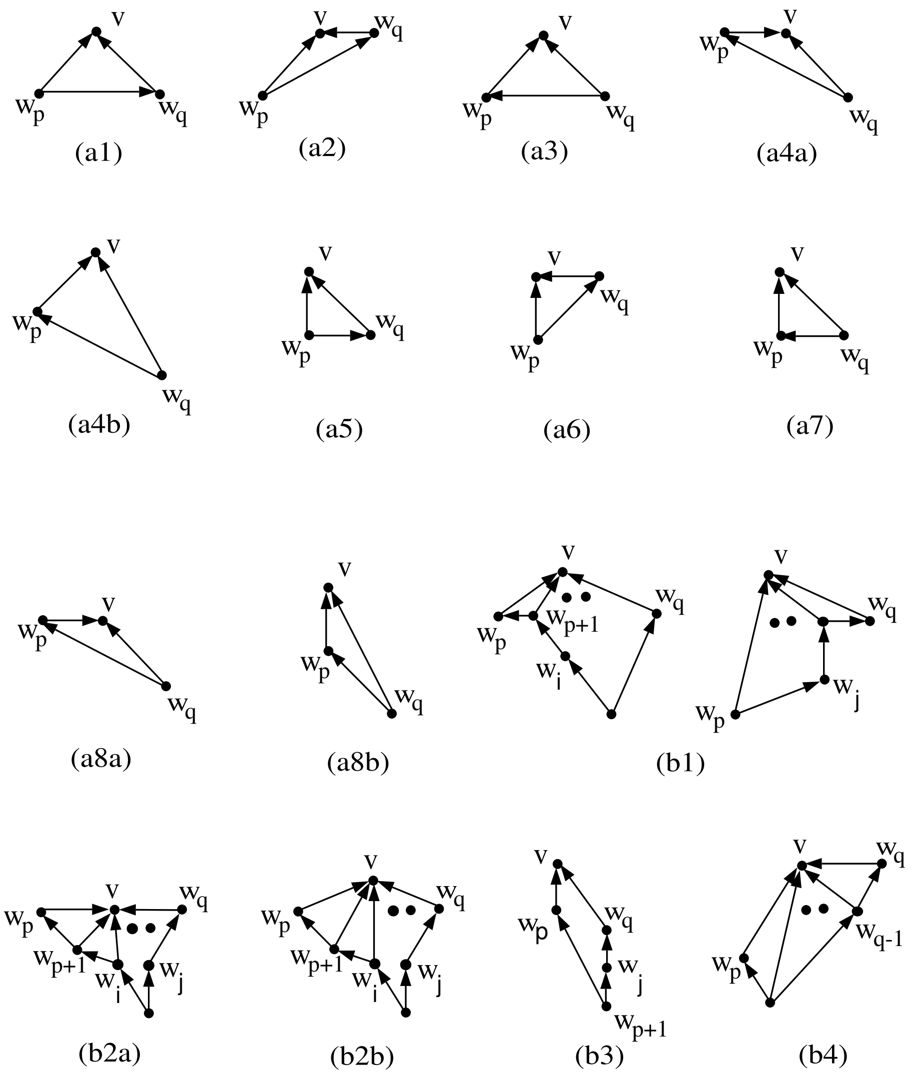

Figure 4: The placement rules.

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 4 (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, the edge (w_p, w_q) is horizontal. v is placed as in Fig 4 (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, the edge (w_p, w_q) is downward. v is placed as in Fig 4 (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 at most 1. H is unchanged.

Case (a8b): Same as case (a8a). Fig 4 (a8b) shows an alternative placement of v: $x(v) = x(w_p)$ and $y(v) = y(w_p) + 1$. W is unchanged. H is increased by at most 1.

We next describe the placement rules for v when $deg^-(v) \geq 3$. By Lemma 3.2 and the contour invariants, v's children $w_p, \ldots, w_q$ 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. (If $y(w_p) = y(w_q)$, then both w_p and w_q are high-end vertices of v). Let (w_i, w_{i+1}) 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 $w_j = w_q$). Note that the last downward edge in P must appear before the first upward or vertical edge in P. So if P has a downward edge, we have $x(w_i) < x(w_j)$. If P has no downward edge, then $w_i = w_p$. By Lemma 3.3, the edge (w_p, w_{p+1}) cannot be vertical. So we also have $x(w_i) < x(w_j)$. Thus we always have $x(w_p) \leq x(w_i) < x(w_j) \leq x(w_p)$. 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 the conditions (pr3), (pr4), and (pr5). In the following placement rules, we will specify $x(v)$, $y(v)$ and show they satisfy the conditions (i), (ii), (pr1) and (pr2). (Thus the placement requirements are satisfied). 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. (This includes the case when both w_p and w_q are high-end vertices of v, and either the edge adjacent to w_p or the edge adjacent to w_q, or both, is horizontal). v is placed as in Fig 4 (b1) (two examples are shown): Let $y(v) = \max\{y(w_p), y(w_q)\} + 1$ and determine $x(v)$ as follows.

- v is stable. Let $x(v) = x(w_i)$. Then we have $x(v) = x(w_i) < x(w_j)$.
- v is unstable. If $w_i = w_p$, let $x(v) = x(w_p) + 1$. If $w_i \neq w_p$, let $x(v) = x(w_i)$.

  If $w_i \neq w_p$, then $x(v) = x(w_i) < x(w_j)$. If $w_i = w_p$, then $x(v) = x(w_p) + 1$. By Lemma 3.3, $x(w_p) < x(w_{p+1}) < x(w_{p+2})$. So the edges $e_1 = (w_p, w_{p+1})$ and $e_2 = (w_{p+1}, w_{p+2})$ cannot be vertical. If e_1 is upward, then $j = p+1$. In this case, $x(v) \leq x(w_{p+1}) = x(w_j) < x(w_{p+2}) \leq x(w_q)$. If e_1 is horizontal or downward, then $j \geq p+2$. In this case, $x(v) \leq x(w_{p+1}) < x(w_{p+2}) \leq x(w_j) \leq x(w_q)$.

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 (this includes the case when both w_p and w_q are high-end vertices of v); and $x(w_j) \geq x(w_p) + 2$. v is placed as in Fig 4 (b2a): Let $y(v) = y(w_p)$ and $x(v) = \max\{x(w_p) + 1, x(w_i)\}$.

If $w_i \neq w_p$, then $x(v) = x(w_i) < x(w_j)$. If $w_i = w_p$, then $x(v) = x(w_p) + 1 < x(w_j)$. W is unchanged. H is unchanged.

Case (b2b): Same as in case (b2a). Fig 4 (b2b) shows an alternative placement of v: Let $y(v) = y(w_p) + 1$, and $x(v) = \max\{x(w_p) + 1, x(w_i)\}$. Similar to case (b2a), we can show $x(v) < x(w_j)$. 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 (this includes the case when both w_p and w_q are high-end vertices of v); and $x(w_j) = x(w_p)+1$. v is placed as in Fig 4 (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. (If v were unstable, the edge (w_{p+1}, w_{p+2}) would not be vertical by Lemma 3.3). Let $y(v) = y(w_p) + 1$ and $x(v) = x(w_p)$. Clearly $x(v) < x(w_j)$. 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 4 (b4): Let $y(v) = y(w_q)$. If v is stable, let $x(v) = x(w_i)$. If v is unstable, let $x(v) = \max\{x(w_p) + 1, x(w_i)\}$. By using the same argument as in case (b1), we can show the conditions (i), (ii), (pr1) and (pr2) are satisfied. W is unchanged. H is unchanged.

This completes the description of the placement rules. In the cases (a4), (a8), and (b2), two alternative placement rules are given. The two alternatives are chosen as follows: Suppose that the vertex $v = v_{k+1}$ $(k + 1 < n - 1)$ satisfies the conditions of the rule (a4) (the rules (a8) or (b2), respectively). Let $w_p, \ldots, w_q$ be the children of v in the contour C_k. Let u be the vertex that covers the edge (w_p, v). Let z_1, z_2, z_3 be the first, the second, and the third child of u, respectively. If the following *avoid-horizontal-forward-edge* conditions hold, we must place v by using the rule (a4b), (the rules (a8b) or (b2b), respectively).

Definition 3.4 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: Note that the rules (a4a), (a8a) and (b2a) are the only rules that can create horizontal forward edges. By using the rules (a4b), (a8b) and (b2b), this can be avoided. More precisely, suppose that v satisfies the avoid-horizontal-forward-edge conditions. 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. This will cause u to be a peak point (see Fig 5(1)). If v is placed by using the rules (a4b), (a8b) or (b2b), the edge (w_p, v) will be either upward or vertical, and v will be a peak point. In this case, u will not be a peak point (see Fig 5 (2)). One might think that using the rules (a4a), (a8a) and (b2a) is better than using the rules (a4b), (a8b) and (b2b), since the placement of v does not increase H. However, the placement of u

will increase H in this case. So using the rules (a4b), (a8b), (b2b) merely shifts the contribution to H from the vertex u to the vertex v. Later, we will see that using the rules (a4b), (a8b) and (b2b) allows us to prove the bound on $W + H$.

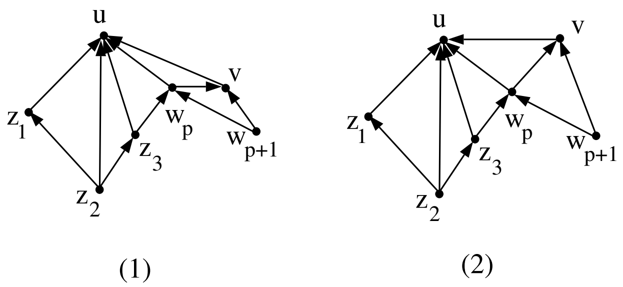

Figure 5: The avoid-horizoantal-forward-edge conditions.

Remark 2: By the definition of π_{left}, the vertices z_1, z_2, z_3 are embedded before v is embedded. 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). In the first case, W is increased by at most 1 and H is not changed. In the second case, W is unchanged and H is increased by at most 1. 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. (n_r can be computed when π_{left} is constructed). Let $d = (n/2 - 1) - n_r$. (Later we will show $d \geq 0$.) 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 shown in Fig 3 is embedded by using our algorithm. (Recall that the vertex $v_n = 19$ is not embedded). The rule used for each vertex is indicated in the figure. The vertex 9 is a free-lance vertex and is placed by using the rule (a8a). The vertex 11 is a (b2b) vertex since it satisfies the avoid-horizontal-forward-edge conditions. Although the vertex 18 satisfies the avoid-horizontal-forward-edge conditions (because of the vertex 19), since 19 needs not be embedded, 18 is placed by using the rule (b2a).

4 Bounding the Grid Size

In this section, we prove the bounds on the size of the grid embedding produced by our algorithm, and complete the proof of Theorem 1.1. We first derive some equations on the grid size. Let W and H be the width and the height of the final grid. If a vertex 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, respectively. 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, respectively. Let a'_i and b'_i be the number of (a'i) and (b'i) vertices, respectively. We have:

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

If the placement of a vertex v increases W, v is called 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. We have:

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

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

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

4.1 Bound on $H + W$

From equations (1), (2), and (3), we have:

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

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

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

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), or (b4), or (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. Then define $x_{i+1} = u$, and continue. Note that the edge (w_p, x_i) is upward. By our placement 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. Note that 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. (See Fig 6(1)). By the avoid-horizontal-forward-edge conditions and the placement rules, the last edge covered by u must be either upward or vertical. Thus, u is placed as 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 6(2)).

- 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.

- 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 the edge e_1 is not horizontal. If e_2 is upward, then neither e_1 nor e_2 is horizontal. So u is placed either as a (b2a), or (b2b), or (b4) vertex. (Since both the first and the second edge covered by u are not vertical, the rule (b3) does not apply).

- 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.

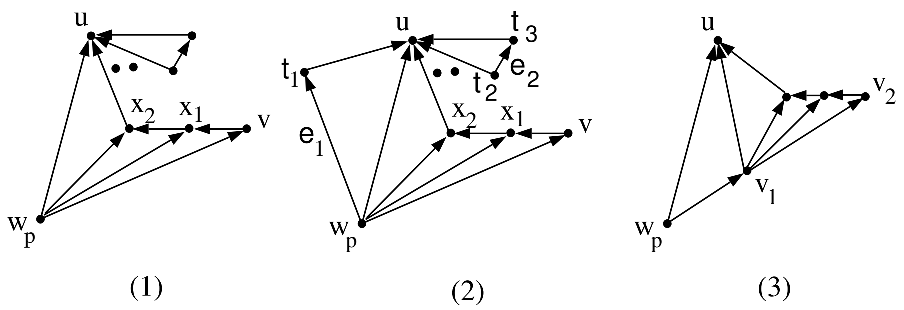

Figure 6: The last vertex u in the sequence $S(v)$.

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 6(3)).

We construct a directed forest F such that the following hold:

- The node set of F is a subset of the vertices of G^+.

- The set of leaf nodes in F is the set D.

- Each non-leaf internal node of F is a (b4) vertex, and has exactly two children.

- 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 $S(v)$. Note that u is either in J or a (b2b) vertex. 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 6 (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 process until all vertices in Q have distinct parents. At this point, 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.

Fig 7 shows an example of this construction. A portion of the graph G and its corresponding forest F is shown. The vertices v_1, v_2, v_3 are in D. The root u_3 is a (b2b) vertex.

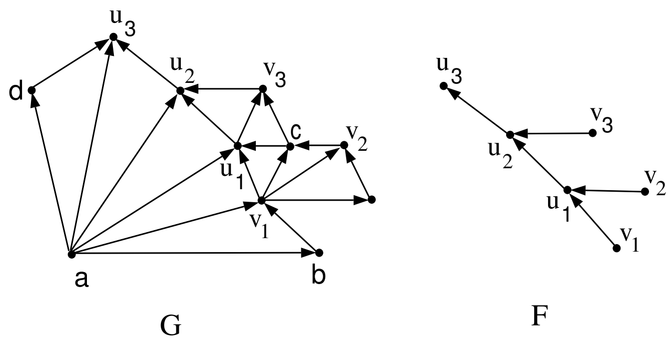

Figure 7: The construction of the forest F.

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. (See Fig 7. The vertex v_1 has the smallest y-coordinate. It is not height-increasing, because there is a vertex z such that $y(z) = y(v_1)$ and $z \prec_{left} d \prec_{left} v_1$). 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: $|D|' \leq |J|$. From the inequality (4), this gives:

$$W + H \leq (n-1) - (a''_1 + a''_2 + a''_3 + a''_{4a} + a''_{4b} + a''_{8a}) \leq (n-1) \quad (5)$$

4.2 Bound on the width W

We first bound n_r (the number of room-shift vertices in G^+). Each room-shift vertex v is associated with a vertex $dom(v)$. It has been shown in [4] that, for two distinct room-shift vertices v_1 and v_2, $dom(v_1) \neq dom(v_2)$. From the definition of the dominator, it is easily seen that 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:

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

4.3 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). Thus

$$a_1 + a_2 + a_3 + a_{4a} + a_{4b} + a_{8a} = n_r + d = n/2 - 1$$

From this, equation (2), and inequality (5), we have:

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

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 8(1)). At most one of x and y can be a height-increasing vertex. Define $mate(v)$ to be the vertex among x and y that is not a height-increasing vertex.

Consider a vertex $v \in B$. Let y and z be the first and the second child of v. Let x be the vertex that covers the edge (y, v). (See Fig 8(2)). By Remark 1, v is a peak point and the edge (x, v) is horizontal. So x is not a height-increasing vertex. Define $mate(v) = x$.

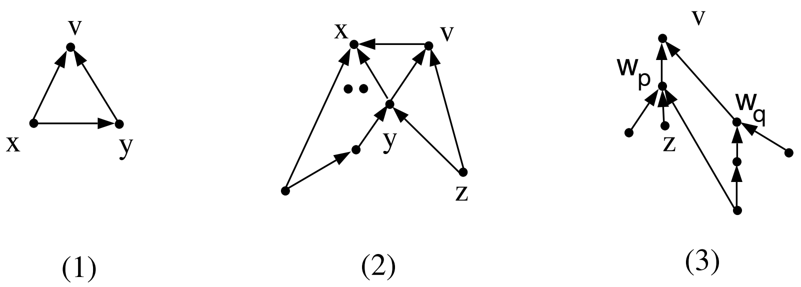

Figure 8: The definition of the mate vertices.

Consider a vertex $v \in C$. Let w_p and w_q be the leftmost and the rightmost child of v, respectively. (See Fig 8(3)). By the rule (b3), we have $y(w_p) \geq y(w_q)$. By the definition of the leftmost canonical ordering 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 a height-increasing vertex. 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:

- If $x \in Mate$, then x is not in $A \cup B \cup C$.
- If $x = mate(v)$ for a vertex $v \in C$, then x cannot be $mate(v')$ for any $v' \neq v$.
- 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 equation (1), this gives:

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

Recall that G^+ has n vertices, and the original graph G has $n - 1$ vertices. So the bounds in inequalities (5), (6), (7), and (8) imply the bounds on grid size stated in Theorem 1.1.

4.4 Implementation of the algorithm

The extended graph G^+ can be constructed from G in linear time. The canonical ordering π in Theorem 3.1 can be computed in $O(n)$ time [14]. The leftmost canonical ordering π_{left} can be computed from π in linear time. After π_{left} is known, the algorithm can be implemented by using the method in [4]. We only note that, as stated in Remark 2, since π_{left} is used, when a vertex v is added, the conditions of the placement rules (including the avoid-horizontal-forward-edge conditions) can be checked in constant time. So the algorithm takes linear time. This completes the proof of Theorem 1.1.

5 Conclusions

We have shown that every internally triangulated plane graph without non-empty triangles has a straight line embedding on a $W \times H$ grid such that $W + H \le n$, $W \le (n+3)/2$ and $H \le 2(n-1)/3$. The embedding can be constructed in linear time.

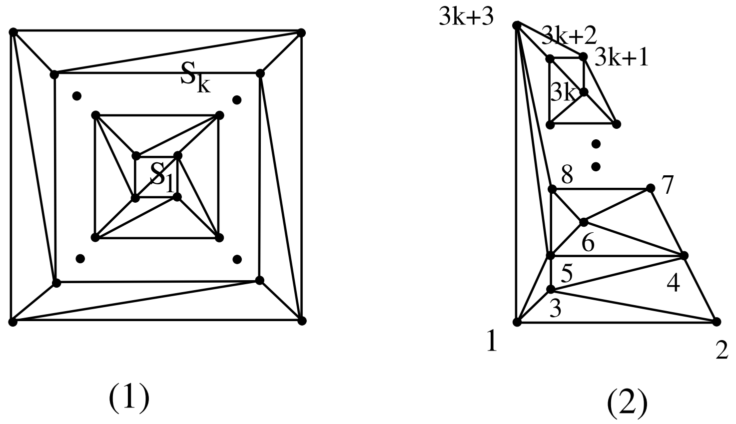

Figure 9: Graphs that require large grids.

Consider the graph G shown in Fig 9 (1). The square S_1 can be embedded in a 1×1 grid. It is easy to show that after S_k has been embedded, we need at least two more x-coordinates and two more y-coordinates to embed the square S_{k+1}. Hence any straight line embedding of G must use a grid of size at least $(n/2-1) \times (n/2-1)$. We conjecture that every internally triangulated plane graph without non-empty triangles can be embedded in an $n/2 \times n/2$ grid. Our algorithm cannot achieve this bound. An example is shown in Fig 9(2). This graph has $3k+2$ vertices connected as shown, where the extra vertex $v_n = 3k+3$ is added and adjacent to every vertex on the external face. Because of the avoid-horizontal-forward-edge conditions, our algorithm places the vertex 4 by the rule (a4b) and the vertices $3i+1$ $(2 \le i \le k)$ by the rule (a8b). The size of the embedding is $(k+2) \times 2k$. (It is easy to embed this graph in an $n/2 \times n/2$ grid, though). It will be interesting to either prove or disprove this conjecture.

References

[1] J. Bondy and U. Murty, Graph Theory with Applications, North-Holland, 1976.

[2] 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.

[3] M. Chrobak and G. Kant, Convex grid drawings of 3-connected planar graphs, Tech. Rep. RUU-CS-93-45, Department of Computer Science, Utrecht University, 1993.

[4] M. Chrobak and S. Nakano, Minimum-width grid drawings of planar graphs, Tech. Rep. UCR-CS-94-5, Department of Computer Science, University of California at Riverside, 1994. Also in Proc. Workshop on Graph Drawing'94, Princeton, NJ, Oct. 1994.

[5] M. Chrobak and T. Payne, A linear time algorithm for drawing planar graphs on a grid, Tech. Rep. UCR-CS-89-1, Department of Mathematics and Computer Science, University of California at Riverside, 1989.

[6] P. Eades and R. Tamassia, Algorithms for automatic graph drawing: an annotated bibliography, Technical Report, Department of Computer Science, Brown University, 1993.

[7] S. Even, Graph Algorithms, Computer Science Press, 1979.

[8] I. Fáry, On straight line representation of planar graphs, Acta. Sci. Math. Szeged, 11, 1948, pp. 229–233.

[9] H. de Fraysseix, J. Pach, and R. Pollack, Small sets supporting straight-line embeddings of planar graphs, in Proc. of the 20th Annual ACM Symp. on Theory of Computing, 1988, pp. 426–433.

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

[11] 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, in Proc. of the 4th Annual ACM Symp. on Parallel Algorithms and Architectures, 1992, pp. 410-419. To appear in SIAM J. Disc. Math.

[12] 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.

[13] G. Kant, Algorithms for drawing planar graphs, Ph.D. Dissertation, Department of Computer Science, University of Utrecht, 1993.

[14] G. Kant and X. He, Two algorithms for finding rectangular duals of planar graphs, in Proc. 19th Workshop on Graph-Theoretic Concepts in Computer Science, Utrecht, the Netherland, June 16-18, 1993.

[15] F. P. Preparata and R. Tamassia, Fully dynamic techniques for point location and transitive closure in planar structures, in Proc. 29th IEEE FOCS, 1988, pp. 558-567.

[16] R. C. Read, A new method for drawing a planar graph given the cyclic order of the edges at each vertex, Congressus Numerantium 56, 1987, pp. 31–44.

[17] P. Rosenstiehl and R. Tarjan, Rectilinear planar layouts and bipolar orientations of planar graphs, Discrete & Computational Geometry 1, 1986, pp. 343–353.

[18] W. Schnyder, Embedding planar graphs on the grid, Abstracts AMS 9, 1988, p. 268.

[19] W. Schnyder, Planar graphs and poset dimension, Orders 5, 1989, pp. 323–343.

[20] W. Schnyder, Embedding planar graphs on the grid, in Proc. of the 1st Annual ACM-SIAM Symp. on Discrete Algorithms, 1990, pp. 138–147.

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

[22] S. K. Stein, Convex maps, in Proc Amer Math Soc, Vol. 2, 1951, pp. 464–466.

[23] W. T. Tutte, How to draw a graph, Proc. London Math. Soc. 13, 1963, pp. 743–768.

[24] K. Wagner, Bemerkungen zum Vierfarben problem, Jahresbericht Deutsch Math-Verein 46, 1936, pp. 26–32.