A Parallel Algorithm for Drawing Planar Graphs on the Grid

2X−2

Liang-Jee JOU, Hitoshi SUZUKI, Takao NISHIZEKI
Tohoku University

1. Introduction

We consider the problem of embedding triconnected cubic planar graphs on a hexagonal grid. The problem is to embed the vertices of such a graph into a hexagonal grid, where the edges lie on grid lines in such a way that all edges except one are straight and any two edges do not intersect.

We first introduce a sequential algorithm, given by Kant [Ka], which embeds a triconnected cubic planar graph of n vertices on an $\frac{n}{2} \times n$ hexagonal grid in $O(n)$ time. We then present a parallel implementation of his algorithm. The parallel algorithm runs in $O(\log n \log^* n)$ time using $O(n)$ processors. Our parallel computation model is CRCW PRAM.

2. Kant's sequential algorithm

The algorithm consists of the following five steps:

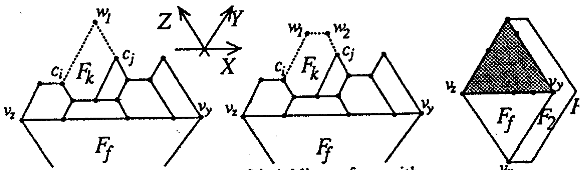
(1) Construct the dual H of a given triconnected cubic planar graph G. Apparently, H is triangular.

(2) Find a canonical numbering of H [FPP]. The f faces F_1, F_2,..., and F_f of G correspond to the f vertices v_1, v_2,..., and v_f of H, respectively. Assume that vertices v_1, v_2, ..., and v_f are indexed according to the canonical numbering.

(3) For each F_i, $3 \le i \le f$, find $E(F_i)$ and $be(F_i)$ defined as follows:
For a face F_i, $3 \le i \le f$, let $E(F_i)$ be the set of edges of F_i which belong to a face F_j such that $j < i$. The *basis-edge* of F_i, $3 \le i < f$, denoted by $be(F_i)$, is the edge $e \in F_i$ that, among all edges in F_i, belongs to the highest numbered face F_j adjacent to F_i. Let $be(F_f)$ be the unique edge $e \in F_f \cap F_1$.

(4) Assign length $lth(e)$ for each basis-edge e in G as follows.
Set $lth(e) := 1$, $\forall e \in G$;
for $k := 3$ to $f-1$ do
  $lth(be(F_k)) := \sum_{e \in E(F_k)} lth(e) - 1$;
$lth(be(F_f)) := \sum_{e \in E(F_f)} lth(e) - 2$;

(5) Draw $F_f, F_{f-1}, \ldots, F_3, F_2, F_1$ sequentially in this order as follows.
Draw F_f as follows: Let v_x be the unique vertex in $F_f \cap F_2 \cap F_1$. Let v_y and v_z be the neighbors of v_x on F_f. We start with drawing v_x on (0,0). From v_x we place v_y $lth(be(F_f))$ steps in Y-direction (see Figure 1) and v_z $lth(be(F_f))$ steps in Z-direction. All other vertices of F_f are placed on the horizontal line segment (of length $lth(be(F_f))$) between v_z and v_y in a way that these horizontal edges e of F_f have length $lth(e)$.
When adding a face F_k by adding vertices and edges of $E(F_k)$ to the current drawing of $F_{k+1}, \ldots, F_f$, we call the added vertices and edges *new*. Let C_{k+1} be the outerface of the current drawing of $F_{k+1}, \ldots, F_f$. Let c_i and c_j ($j > i$) be the two vertices of C_{k+1}, to which new edges of F_k are incident, then we call c_i the *startpoint* and c_j the *endpoint* of face F_k, respectively.
Adding a face goes as follows: if we add exactly one vertex then we walk from c_i upwards in Y-direction and from c_j upwards in Z-direction. The crossing point is the place for the new vertex. If we add two or more vertices $w_1, \ldots, w_p$ ($p \ge 2$), then we go from c_j one unit in Y-direction and from c_i in Z-direction to the same height (assume $y(c_j) \ge y(c_i)$) and place the new vertices on the horizontal line segment between them. Face F_2 is drawn as illustrated in Figure 1(c). Face F_1 is the outer face.



(a) Adding a face with one vertex to the current drawing. (b) Adding a face with two vertices to the current drawing. (c) The complete drawing.

Figure 1

Each of steps 1, 3, 4 and 5 can be executed in $O(n)$ time. Step 2 also can be executed in $O(n)$ time [CP]. Thus the sequential algorithm runs in $O(n)$ time.

Theorem 2.1 [Ka] There is an $O(n)$ time algorithm tc embed any triconnected cubic planar graph on an $\frac{n}{2} \times n$ hexagonal grid such that all edges except one are straight.

3. Parallel implementation

Our parallel algorithm is as follows:

(1) Construct the dual H of a given triconnected cubic planar graph G. Apparently, H is triangular.

(2) Construct a *realizer* of the triangular graph H [Sch]

(3) For each interior vertex v of H, find $be(F)$ and $E(F)$ where F is the face of G corresponding to v. Fo convenience, we call such a face F an *interior face* o G.

(4) For each interior face F of G, calculate $lth(be(F))$ Also calculate $lth(be(F_f))$.

(5) For each interior face F, calculate the X and Y co ordinates for all of its new vertices.

We then analyze the correctness and time-complexity.

Step (1) can be executed in $O(\log n)$ time with $O(n$ processors [GR].

In step (2), we construct a realizer (instead of a canonical numbering) of the triangular graph H, whicl is defined in the following definition [Sch]. This step car be executed in $O(\log n \log^* n)$ time with $O(n)$ processor: [He].

Definition 3.1 A *realizer* of a triangular graph H is a par tition of the interior edges of H into three sets $\{T_1, T_2, T_n$ of directed edges of trees such that the following hold.

(1) For each interior vertex v, the edges incident with v appear around v in counterclockwise order as follows

---

L.J. Jou, H. Suzuki, T. Nishizeki, Department of Information Engineering, Faculty of Engineering, Tohoku University, Sendai 980, Japan.

one edge in T_1 leaving v; a set (maybe empty) of edges in T_n entering v; one edge in T_2 leaving v; a set (maybe empty) of edges in T_1 entering v; one edge in T_n leaving v; a set (maybe empty) of edges in T_2 entering v.

(2) Let v_1, v_2 and v_n be the three exterior vertices of H appearing in counterclockwise order. All interior edges incident with v_1, v_2 and v_n enter v_1, v_2 and v_n, respectively, and belong to T_1, T_2 and T_n, respectively.

Theorem 3.1 [Sch] Let H be a triangular graph with at least four vertices. Then H has a realizer $\{T_1, T_2, T_n\}$. Moreover, each T_i ($i = 1,2,n$) is a tree including all interior vertices and exactly one exterior vertex v_i, and all edges of T_i are directed toward v_i.

Each interior vertex v of H has three neighbors x, y and z such that edges (v,x), (v,y) and (v,z) leave v and are in T_1, T_2 and T_n, respectively. Denote x, y and z by $T_1(v)$, $T_2(v)$ and $T_n(v)$, respectively. We then have the following two lemmas.

Lemma 3.2 Given a realizer of a triangular graph H, one can construct a canonical numbering of H such that, for each interior vertex v of H, the neighbors of v appearing around v between $T_1(v)$ and $T_2(v)$ in counterclockwise order (including $T_1(v)$ and $T_2(v)$) have indices less than $ind(v)$, and the other neighbors of v have indices greater than $ind(v)$.

Proof. Omitted.

Lemma 3.3 For each interior vertex v, $T_n(v)$ has the greatest index among the neighbors of v.

Proof. Omitted.

Using Lemmas 3.2 and 3.3, we can show that Step (3) can be done efficiently in parallel as follows.

Lemma 3.4 Let G be a triconnected cubic planar graph and H the dual. Given a realizer of H, one can find $be(F)$ and $E(F)$ in parallel for interior faces F of G. It takes $O(\log n)$ time with $O(n)$ processors.

We implement step (4) as follows.

(4-1) Construct a tree T_{be} defined as follows: T_{be} is a rooted tree consisting of $f-2$ nodes.
  (a) the root node corresponds to F_f;
  (b) each non-root node of T_{be} corresponds to an interior face of G;
  (c) node n_k is the parent of node n_j in T_{be} if $be(F_j)$ is in F_k, where F_k and F_j are faces of G corresponding to n_k and n_j, respectively.

(4-2) Calculate $lth(be(F))$ and $lth(be(F_f))$ by using the doubling technique for T_{be}.

Step (4) can be executed in $(\log n)$ time with $O(n)$ processors.

We implement step (5) as follows.

(5-1) For each interior face F of G, find startpoint c_i and endpoint c_j of F.

(5-2) For each interior face F of G, calculate the X-coordinates of its new vertices $w_1,...,w_p$.

(5-3) For each interior face F of G, calculate the Y-coordinates of its new vertices $w_1,...,w_p$.

Clearly, using $O(n)$ processors, step (5-1) and (5-2) can be executed in $O(1)$ and $O(\log n)$ time, respectively. Therefore we shall show how to execute step (5-3) efficiently in parallel.

(5-3-1) Construct trees T_{c_j} and T_{c_i}, which are defined as follows.
  T_{c_j} is a rooted tree consisting of $f-2$ nodes:
  (a) the root node corresponds to F_f;
  (b) each non-root node corresponds to an interior face of G;
  (c) node n_{k_1} is the parent node of node n_{k_2} in T_{c_j} if the endpoint c_j of F_{k_2} is a new vertex of F_{k_1}.
  T_{c_i} is a rooted tree constructed from T_{c_j} as follows: for every two nodes n_{k_1} and n_{k_2} of T_{c_j}, add to T_{c_j} an edge directed from n_{k_2} to n_{k_1} and delete from T_{c_j} the edge directed from n_{k_2} to its parent if (1) the startpoint c_i of F_{k_2} is a new vertex of F_{k_1} and (2) F_{k_2} has two or more new vertices.

We then have the following lemma.

Lemma 3.5 For each interior face F of G and its corresponding c_i and c_j, the vertex $u \in \{c_i, c_j\}$ having higher Y-coordinate can be known by using T_{c_i} and T_{c_j} in $O(\log n)$ time with $O(n)$ processors.

Proof. Omitted.

(5-3-2) Construct a tree $T_{c_{ij}}$, which is defined as follows.
  $T_{c_{ij}}$ is a rooted tree consisting of $f-2$ nodes:
  (a) the root node corresponds to F_f;
  (b) each non-root node corresponds to an interior face of G;
  (c) node n_{k_1} is the parent node of node n_{k_2} in $T_{c_{ij}}$ if for face F_{k_2}, the vertex $u \in \{c_i, c_j\}$ having higher Y-coordinate is a new vertex of F_{k_1}.

(5-3-3) For each interior face F of G, calculate the Y-coordinates of F's new vertices by using the doubling technique for $T_{c_{ij}}$.

Hence step (5) can also be executed in $O(\log n)$ time with $O(n)$ processors.

We thus can conclude the following theorem.

Theorem 3.6 There is a parallel algorithm which embeds a triconnected cubic planar graph on an $\frac{n}{2} \times n$ hexagonal grid in $O(\log n \log^* n)$ time with $O(n)$ processors.

References

[CP] M. Chrobak and T. H. Paynes, A linear time algorithm for drawing planar graphs on the grid, Tech. Rep. UCR-CS-90-2, Department of Mathematics and Computer Science, University of California at Riverside, 1990.

[FPP] H. de Fraysseix, J. Pach, R. Pollack, Small sets supporting Fáry embeddings of planar graphs, Proc. 20th Ann. ACM Symp. on Theory of Computing, 426-433, 1988.

[GR] A. Gibbons and W. Rytter, Efficient parallel algorithms, Cambridge Univ. Press, Cambridge, 1987.

[He] X. He, Efficient parallel algorithms for two graph layout problems, Tech. Rep. 91-05, Department of Computer Science, State University of New York at Buffalo, 1991.

[Ka] G. Kant, Hexagonal grid drawings, Tech. Rep. CS-92-06, Department of Computer Science, Utrecht University, 1992.

[Sch] W. Schnyder, Embedding planar graphs on the grid, Proc. First Annual ACM-SIAM Symp. on Discrete Algorithms, San Francisco, pp. 138-147, 1990.