<table>
<thead>
<tr>
<th>Title</th>
<th>A Simple Near Optimal Parallel Algorithm for Recognizing Outerplanar Graphs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Author(s)</td>
<td>Nakayama, Shin-ichi; Masuyama, Shigeru</td>
</tr>
<tr>
<td>Citation</td>
<td>数理解析研究所講究録 (1995), 906: 6-13</td>
</tr>
<tr>
<td>Issue Date</td>
<td>1995-04</td>
</tr>
<tr>
<td>URL</td>
<td><a href="http://hdl.handle.net/2433/59467">http://hdl.handle.net/2433/59467</a></td>
</tr>
<tr>
<td>Type</td>
<td>Departmental Bulletin Paper</td>
</tr>
<tr>
<td>Textversion</td>
<td>publisher</td>
</tr>
</tbody>
</table>

Kyoto University
A Simple Near Optimal Parallel Algorithm for Recognizing Outerplanar Graphs

Shin-ichi Nakayama (中山 慎一)  Shigeru Masuyama (増山 繁)

Department of Knowledge-Based Information Engineering, Toyohashi University of Technology Toyohashi-shi, Aichi 441, Japan
E-mail: shin@toki.tutkie.tut.ac.jp, masuyama@tutkie.tut.ac.jp

Abstract. An outerplanar graph is a graph which can be embedded in the plane so that all vertices lie on the boundary of the exterior face. In this paper, we propose a simple near optimal parallel algorithm for recognizing whether a given graph $G$ is outerplanar in $O(\log n)$ time using $O(n\alpha(l,n)/\log n)$ processors on an arbitrary-CRCW PRAM where $n$ is the number of vertices in $G$, $\alpha(l,n)$ is the inverse Ackermann function, which grows extremely slowly with respect to $l$ and $n$ [9] and $l = O(n)$. Although a near optimal parallel algorithm for general graphs can also be obtained by combining the algorithm in [3] with the algorithm for finding biconnected components [4][9], our algorithm uses methods completely different from the algorithm in [3]'s and is much simpler than [3]'s.

1 Introduction

An outerplanar graph is an undirected graph which can be embedded in the plane in such a way that all vertices lie on the exterior face (see Fig. 1). A graph always denotes an undirected graph throughout this paper, except when it is specified to be directed. For outerplanar graphs, several efficient algorithms for solving important problems e.g., vertex-coloring, edge-coloring, longest path, are known [9][5]. Furthermore, it is well-known that a given graph is outerplanar if and only if a given graph has page number one, where graph $G$ has page number one if there exists a linear arrangement of vertices so that no pair of edges is crossing when they are drawn on the same side of the linear arrangement of the vertices [13][11]. The problem of deciding whether a given graph has page number one is the special case of the book embedding, whose application to fault-tolerant VLSI design is described e.g., in the introduction of [13]. Thus, it is useful to develop efficient algorithms for recognizing whether a given graph is outerplanar or not.

Mitchell [10] proposed an $O(n)$ sequential algorithm for recognizing outerplanar graphs where $n$ is the number of vertices in $G$. The sequential algorithm removes a vertex $v$ satisfying some properties from a given graph $G$ step by step, and cannot straightforwardly be applied to develop an efficient parallel algorithm. Diks, Hagerup and Rytter [3] developed a parallel algorithm for recognizing outerplanar graphs. When an input graph is biconnected, the algorithm [3] runs in $O(\log n)$
time using $O(n/\log n)$ processors on a CRCW PRAM (see e.g., [8]), where $n$ is the number of vertices in $G$. However, when an input graph is a general graph, we need to find biconnected components before applying the algorithm [3] to each biconnected component. The best known parallel algorithm for finding biconnected components runs in $O(\log n)$ time using $O((n + m)\alpha(m, n)/\log n)$ processors on the arbitrary-CRCW PRAM [4] [9] where $m$ is the number of edges and $\alpha(m, n)$ is the inverse Ackermann function, which grows extremely slowly with respect to $m$ and $n$ [9]. \footnote{If the class of input graphs is \textit{linearly contractible graph class} [7] such as the class of planar graphs, an optimal parallel algorithm for finding biconnected components that runs in $O(\log n)$ time using $O(n/\log n)$ processors on the arbitrary-CRCW PRAM exists [7]. However, this algorithm does not work for general graphs.} The arbitrary-CRCW PRAM is defined by the property that when several processors try to write to the same memory cell in the same step, then exactly one of them succeeds [8]. As outerplanar graphs have at most $2n - 3$ edges [10], by checking this fact first, we can find biconnected components in $O(\log n)$ time using $O(n\alpha(l, n)/\log n)$ processors on the arbitrary-CRCW PRAM where $l = O(n)$. Thus, the algorithm [3] combined with the algorithm for finding biconnected components [4] [9] takes, in total, $O(\log n)$ time using $O(n\alpha(l, n)/\log n)$ processors on the arbitrary-CRCW PRAM, when applied to general graphs. Similarly, on a CREW PRAM (see e.g., [8]), the complexity of parallel algorithm [3] is dominated by finding biconnected components, when applied to general graphs.

In this paper, we present a simple near optimal parallel algorithm for recognizing outerplanar graphs in $O(\log n)$ time using $O(n\alpha(l, n)/\log n)$ processors on the arbitrary-CRCW PRAM, in the sense that $O(\log n) \times O(n\alpha(l, n)/\log n) = O(n\alpha(l, n))$ is almost linear with respect to $n$. Although a near optimal parallel algorithm for general graphs can also be obtained by combining the algorithm in [3] with the algorithm in [4] [9], our algorithm uses methods completely different from the algorithm in [3]'s, e.g., the well known $st$-numbering, and is much simpler than [3]'s.

## 2 Definitions

Given an undirected connected graph $G = (V, E)$ having no multiple edges. A path $P$ from $v_0$ to $v_k$ in $G$ is a finite non-null sequence $v_0, e_1, v_1, e_2, v_2, \ldots, e_k, v_k$, $v_i \in V$, $i = 0, 1, \ldots, k$, $e_j \in E$, $j = 1, 2, \ldots, k$, such that, for $1 \leq i \leq k$, the end vertices of $e_i$ are $v_{i-1}$ and $v_i$, respectively. If $v_0 = v_k$, then path $P$ is a circuit.

A biconnected graph $G$ is a connected graph which has no vertex $v$ such that $G - v$ (the graph obtained by removing $v$ from $G$) has at least two connected components. A biconnected outerplanar graph has a planar embedding consisting of a circuit bounding the exterior face, where (possibly) a number of non-crossing edges are embedded within the interior region of this circuit [5]. Edges on the boundary of the exterior face are called sides, while the other edges are called diagonals [5].

Next, we describe the $st$-numbering used in our parallel algorithm.

**Definition 1** [12] An $st$-numbering is a one-to-one function $f$ from $V$ to $\{1, \ldots, n\}$ satisfying the following two conditions:

(i) $f(s) = 1$ and $f(t) = n$, 

(ii) for each \( v \in V - \{s,t\} \), there exist adjacent vertices \( v_1 \) and \( v_2 \) such that \( f(v_1) < f(v) < f(v_2) \).

Fig. 2 illustrates \( st \)-numbering. The \( st \)-numbering is used as an indispensable component in several algorithms [12]. We have the following theorem.

**Theorem 1** [12] A graph \( G \) is biconnected if and only if it has an \( st \)-numbering by letting \( s = u \) and \( t = v \) for each edge \((u,v)\).

(Note 2.1) If graph \( G \) is biconnected, its \( st \)-numbering can be obtained in \( O(\log n) \) time using \( O((n + m)\alpha(m,n)/\log n) \) processors [4] where \( n \) (resp., \( m \)) is the number of vertices (resp., edges) in \( G \) and \( \alpha(m,n) \) is the inverse Ackermann function.

### 3 The Parallel Algorithm

We first assume that the given graph \( G \) is biconnected. We shall describe how to treat general graphs at the end of this section. The following theorems characterize outerplanar graphs.

**Theorem 2** [6] Given graph \( G = (V,E) \), \( G \) is outerplanar if and only if \( G \) has no subgraph homeomorphic to either \( K_4 \) or \( K_{2,3} \), where \( K_4 \) is the complete graph on four vertices and \( K_{2,3} \) is the graph illustrated in Fig. 3. \( \square \)

**Theorem 3** [10] An outerplanar graph \( G \) with \( n(\geq 3) \) vertices has

(i) at most \( 2n - 3 \) edges,

(ii) at least two vertices of degree 2. \( \square \)

Our parallel algorithm first checks, based on Theorem 3, if \( G \) has at most \( 2n - 3 \) edges and at least two vertices of degree 2. Then, this algorithm chooses a vertex \( v \) of degree 2 and a vertex \( v' \) incident to \( v \); regards \( v \) (resp., \( v' \)) as \( s \) (resp., \( t \)) and finds \( st \)-numbering of \( G \). Note that, by Note 2.1 just after Theorem 1, we can find \( st \)-numbering of \( G \) because \( G \) is assumed to be biconnected. When \( G \) is outerplanar, exactly one Hamiltonian circuit always exists in \( G \), and the edges constructing the Hamiltonian circuit can be regarded as sides of the outerplanar graph [2][5]. Consequently, the above process finds the sides by the following lemma. In the following, suppose that the vertices in \( G \) are numbered from 1 to \( n \) by \( st \)-numbering where \( s \) is a vertex of degree 2 and \( t \) is a vertex incident to \( s \) and each vertex in \( G \) is identified with its vertex number.

**Lemma 1** If \( G \) is outerplanar, then all edges \((i, i + 1), i = 1, \ldots, n - 1, \) are in \( G \).

(proof) We shall show that, if \( G \) does not have some edge among \((i, i + 1), i = 1, \ldots, n - 1, \) then \( G \) is not outerplanar. Assume that vertex \( i \) is not incident to vertex \( i+1 \). By the definition of \( st \)-numbering, each vertex \( x, x = 2, \ldots, n - 1, \) must be incident to a vertex whose number is less than \( x \) and to a vertex whose number is more than \( x \), respectively. By this fact and the connectivity of \( G \), \( G \) has simple path \( P_{i,s} \) \( i, j_1, j_2, \ldots, j_l, s, \) \( (l \geq 1) \) where \( i > j_1 > j_2 > \cdots > j_l > 1 (= s) \). Vertex 1 \((=s)\) is adjacent to exactly two vertices \( n (= t) \) and 2 by definition, so \( j_l \) of \( P_{i,s} \) must be 2, (see Fig. 4). Similarly, for \( i+1, \) simple path \( P_{i+1,s} = i + 1, j'_1, j'_2, \ldots, j'_l, s, \) \( (l' \geq 1) \) where \( i + 1 > j'_1 > j'_2 > \cdots > 2 (= j'_l) > 1 (= s) \) exists. Moreover, by the fact that each vertex \( x, x = 2, \ldots, n - 1, \) must be incident to the vertex
whose number is more than \( x \), \( G \) has simple paths \( P_{t,t} = i, k_1, k_2, \cdots, t \), where \( i < k_1 < k_2 < \cdots < t (= n) \), and \( P_{i+1,t} = i + 1, k'_1, k'_2, \cdots, t \), where \( i + 1 < k'_1 < k'_2 < \cdots < t (= n) \).

Since \( t > \cdots > k_2 > k_1 > i > j_1 > j_2 > \cdots > j_i > 1 (= s) \), \( P_{t,t} \) and \( P_{i,s} \) share no vertex except \( i \). Similarly, \( P_{i,t} \) and \( P_{i+1,s}, P_{i+1,t} \) and \( P_{i,s}, P_{i+1,t} \) and \( P_{i+1,s} \) share no vertex except \( i \), \( i + 1 \). \( G^* \), constructed by \( P_{i,s}, P_{i+1,s}, P_{i,t} \) and \( P_{i+1,t} \), has a subgraph homeomorphic to \( K_{2,3} \) (see Fig 4). Hence, \( G \) is not outerplanar by Theorem 2, which however contradicts the assumption that \( G \) is outerplanar. Thus we have shown that if \( G \) is outerplanar, then \( G \) has all edges \( (i, i + 1), i = 1, \cdots, n - 1. \)

By Lemma 1, if at least one edge among \( (i, i + 1), i = 1, \cdots, n - 1 \), does not exist in \( G \), then the algorithm stops since \( G \) is not outerplanar, otherwise the edges \( (i, i + 1), i = 1, \cdots, n - 1 \), and \( (n, 1) \) construct a Hamiltonian circuit \( C \). We regard the edges constructing \( C \) as sides of the outerplanar graph. (Note that if \( G \) is outerplanar, Hamiltonian circuit \( C \) is unique [5].)

We assume that \( G \) is embedded in the plane so that each edge of \( C \) bound the exterior face and the edges of \( G - C \) (\( G - C \) denotes the graph obtained by removing edges of \( C \) from \( G \)) are embedded within the interior region of \( C \). The edges of \( G - C \) are called \textit{diagonals} of \( G \). If the diagonals do not intersect each other on such embedded edges, then \( G \) is outerplanar, otherwise \( G \) is not outerplanar.

To see this, we execute the following process. Hereafter, we identify each vertex with its vertex number assigned by \textit{st-numbering}.

Let \( M(i), i = 1, \cdots, n \), be an array such that \( M(i) \) contains vertex \( j_0 \) where \( j_0 \equiv \min \{ j \mid j \) is the endpoint of diagonals adjacent to \( i \} \).

If there is no diagonal incident to \( i \), \( M(i) \) has a value \( +\infty \) where \( +\infty \) is a sufficiently large number satisfying \( +\infty > n \). For each diagonal \( (x, y) \) such that \( x < y \), we execute \( \text{val}(x, y) \leftarrow \min \{ M(i) \mid x \leq i \leq y \} \) and regard \( \text{val}(x, y) \) as the value of diagonal \( (x, y) \). On the value \( \text{val}(x, y) \) for each diagonal \( (x, y) \), we obtain the following lemma.

**Lemma 2** Assume that Hamiltonian circuit \( C \) is embedded in the plane so that each edge of \( C \) bounds the exterior face and diagonals are embedded within the interior region of \( C \).

The diagonals intersect each other if and only if there is a diagonal \( (x, y) \), where \( x < y \), such that the value \( \text{val}(x, y) \) is less than vertex number \( x \).

(proof) \( (\Rightarrow) \) Assume that there is a pair of diagonals which intersect each other. Let \( (x, y), (x', y') \), where \( x < y, x' < y' \) and \( x' < x \), be a pair of intersecting diagonals. As these two diagonals intersect each other, vertex \( y' \) satisfies \( x < y' < y \) and is adjacent to diagonal \( (x', y') \) where \( x' < x \) (See Fig. 6(a)). Hence, \( \text{val}(x, y) = \min \{ M(i) \mid x \leq i \leq y \} < x \).

(\( \Leftarrow \)) Assume that no diagonals intersect each other. Since no diagonals intersect each other, each vertex \( j \) adjacent to vertex \( i \), where \( x \leq i \leq y \), satisfies \( x \leq j \leq y \) for each diagonal \( (x, y) \) where \( x < y \) (See Fig. 6(b)). Hence, \( \text{val}(x, y) = \min \{ M(i) \mid x \leq i \leq y \} \geq x \).

In the following, we introduce **Procedure Recognition** for recognizing whether a given graph is outerplanar.
Procedure Recognition
begin
(Step 1) if \( m > 2n - 3 \),
then print "G is not outerplanar" and stop.
(Step 2) if \( G \) does not have at least two vertices of degree 2,
then print "G is not outerplanar" and stop.
(Step 3) Choose a vertex \( v \) of degree 2 and a vertex \( v' \) incident to \( v \); regard \( v \) and \( v' \) as \( s \) and \( t \), respectively, and find an st-numbering of \( G \) [12][4].
(Step 4) if \( G \) does not have at least one edge among \( (i, i+1) \) for all \( i, 1 \leq i \leq n - 1 \), where \( i, i+1 \) are the vertex numbers assigned by Step 3,
then print "G is not outerplanar" and stop.
(Step 5) For each vertex \( i, i = 1, \cdots, n \),
\[ M(i) \leftarrow \min \{ j \mid j \text{ is the endpoint of diagonals adjacent to} i \} \]
(Step 6) For each diagonal \( e_j = (x, y) \) where \( x < y \),
\[ \text{val}(x, y) \leftarrow \min \{ M(i) \mid x \leq i \leq y \} \]
(Step 7) if there is a diagonal \( (x, y) \), where \( x < y \), such that \( \text{val}(x, y) < x \),
then print "G is not outerplanar",
else print "G is outerplanar".
end. □

The correctness of Procedure Recognition is obvious by Theorem 3 and Lemmas 1 and 2. We then analyze the computation time and the number of processors required.

The complexity analysis is done under the assumption that each vertex of the input graph \( G \) has a pointer to its predefined adjacency list, that is, for each vertex \( v \in V \), the vertices adjacent to vertex \( v \) are given in a linked list, say, \( L[v] = \langle u_1, u_2, \cdots, u_d \rangle \), in some order, where \( d \) is the degree of \( v \) (Fig. 5(a)). Recall that the arbitrary-CRCW PRAM is used as a parallel computation model in this paper.

The list ranking algorithm [8] can handle steps 1, 2 in \( O(\log n) \) time using \( O(n/\log n) \) processors.

Note that \( m = O(n) \) in the following analysis, as steps 3-7 are executed only when \( m \leq 2n - 3 \) by step 1.

The parallel algorithm for finding st-numbering runs in \( O(\log n) \) time using \( O((n+m)\alpha(m, n)/\log n) \) processors [4] where \( n \) (resp., \( m \)) is the number of vertices (resp., edges) in input graphs and \( \alpha(m, n) \) is the inverse Ackermann function. Thus, in step 3, finding st-numbering of \( G \) requires \( O(\log n) \) time using \( O(n\alpha(l, n)/\log n) \) processors where \( l = O(n) \).

After finding the st-numbering, each of the initial vertex numbers in the adjacency lists \( L[i]'s \) is replaced by its number assigned by the st-numbering. For this process, we first transform the adjacency lists \( L[i]'s \) into a linked list \( L' \) as follows. Let a vertex \( u_d \) be the last element in the adjacency list \( L[i] \) of vertex \( i \) and a vertex \( u_0 \) the first element in \( L[i+1] \). Each vertex \( u_{d} \) has a pointer to \( u_{d+1} \), for \( i = 1, \cdots, n - 1 \), (See Fig. 5(b)). We then convert the linked list \( L' \) into an array \( A \) by applying the list ranking algorithm [8] which runs in \( O(\log n) \) time using \( O(n/\log n) \) processors. And we replace each of the initial vertex numbers by its number assigned by st-numbering using a standard technique used to implement Brent's scheduling principle[5][8] as follows. Partition elements of \( A \) into equal-sized blocks \( E_i, i = 1, \cdots, |A|/\log n \), where each size is \( O(\log n) \). Treat each block \( E_i \) separately, and sequentially replace each of the initial vertex numbers belonging to block \( E_i \) by its number assigned by st-numbering. This process runs in \( O(\log n) \) time using \( O(n/\log n) \) processors.
Step 4 runs in $O(\log n)$ time using $O(n/\log n)$ processors by applying Brent’s scheduling principle[5][8] stated in step 3.

Let $A[k, k']$, $1 \leq k < k' \leq |A| (= O(n))$ be an interval between $k$ and $k'$ in $A$. Note that the elements in $A$ are numbers assigned by st-numbering. As the degree of each vertex is found in step 2, we can recognize the vertices adjacent to vertex $v$ as the element in interval $A[k, k']$ where $1 \leq k < k' \leq |A|$. For example, assume that $d_i$ is the degree of vertex $i$, the vertices adjacent to vertex 1 are the elements in $A[1, d_1]$, the vertices adjacent to vertex 2 are the elements in $A[d_1 + 1, d_1 + d_2]$, and so on. (Note: Given the degree of each vertex, the intervals in $A$ corresponding to vertex $i$ for $i = 1, \cdots, n$, are found in $O(\log n)$ time using $O(n/\log n)$ processors by applying prefix sums algorithm [8].) Hence, in step 5, finding each minimum vertex number adjacent to vertex $i$ for $i = 1, \cdots, n$, can be done by computing the minimum of interval in $A$ corresponding to vertex $i$. As described in [8][pp. 131-136], after executing a preprocessing algorithm (ALGORITHM 3.8 in [8]) which runs in $O(\log n)$ time using $O(n/\log n)$ processors, we can compute the minimum $A_{\text{min}}[k_i, k'_i]$ of $A[k_i, k'_i]$, that is, $\min\{A(k_i), A(k_i + 1), \cdots, A(k'_i)\}$, where $1 \leq k_i < k'_i \leq |A|$, in $O(1)$ time using $O(1)$ processors. We need to compute the minimum $A_{\text{min}}[k_i, k'_i]$’s corresponding to vertex $i$, $i = 1, \cdots, n$. Hence, by Brent’s scheduling principle[5][8], we can compute the minimum $A_{\text{min}}[k_i, k'_i]$’s for $i = 1, \cdots, n$, in $O(\log n)$ time using $O(n/\log n)$ processors. The total complexity in step 5 is $O(\log n)$ time using $O(n/\log n)$ processors.

In step 6, we compute $\min\{ M(i) | x \leq i \leq y \}$, where $x < y$, for each diagonal $e_j = (x, y)$, $j = 1, \cdots, k (= O(n))$. Since this process is equivalent to the process described in step 5, this can be done in $O(\log n)$ time using $O(n/\log n)$ processors.

Step 7 takes $O(\log n)$ time using $O(n/\log n)$ processors.

Having assumed that the input graph $G$ is a biconnected graph so far, we shall describe, before closing this section, how to decide whether $G$ is outerplanar when $G$ is a general graph. We first check if $G$ has at most $2n - 3$ edges. We next find biconnected components, that is, blocks $B_1, B_2, \cdots, B_k$ of $G$ by applying the algorithm of finding biconnected components in [4][9], which runs in $O(\log n)$ time using $O(n\alpha(l, n)/\log n)$ processors. If $G$ is outerplanar, then each of blocks $B_1, B_2, \cdots, B_k$ is also outerplanar [2]. Thus, we independently execute Procedure Recognition for each of these blocks $B_1, B_2, \cdots, B_k$. If a block $B_i$ is an edge, then Procedure Recognition tells that $B_i$ is outerplanar. When each block $B_i$, $i = 1, \cdots, k$, is outerplanar, we print “$G$ is outerplanar” and stop. By the above-mentioned statements, we have the following theorem.

Theorem 4 Given a graph $G$ with $n$ vertices and $m$ edges, whether $G$ is outerplanar or not can be decided in $O(\log n)$ time using $O(n\alpha(l, n)/\log n)$ processors on the arbitrary-CRCW PRAM where $\alpha(l, n)$ is the inverse Ackermann function, which grows extremely slowly with respect to $l$ and $n$ [9] and $l = O(n)$. □
References


Figure 1: An example of an outerplanar graph.

Figure 2: An example of st-numbering.
図 3: $K_{2,3}$.

図 4: Illustration of the proof of Lemma 1.

図 5: Adjacency lists $L(i)$, $i = 1, \cdots, n$, and linked list $L'$.

図 6: Illustration of the proof of Lemma 2.