Parallel Algorithms for Hamiltonian 2-Separator Chordal Graphs

B.S. Panda
Department of Computer and Information Sciences
University of Hyderabad, Hyderabad - 500 046, INDIA
E-mail: bspcs@uohyd.ernet.in

Vijay Natarajan
Department of Computer Science
Duke University, Durham, USA
E-mail: vijay@cs.duke.edu

Sajal K. Das
Department of Computer Science and Engineering
The University of Texas at Arlington
Arlington, TX 76019-0015, USA
E-mail: das@cse.uta.edu

Abstract

In this paper we propose a parallel algorithm to construct a one-sided monotone polygon from a Hamiltonian 2-separator chordal graph. The algorithm requires $O(\log n)$ time and $O(n)$ processors on the CREW PRAM model, where n is the number of vertices and m is the number of edges in the graph. We also propose parallel algorithms to recognize Hamiltonian 2-separator chordal graphs and to construct a Hamiltonian cycle in such a graph. They run in $O(\log^2 n)$ time using $O(mn)$ processors on the CRCW PRAM model and $O(\log^2 n)$ time using $O(m)$ processors on the CREW PRAM model, respectively.

Keywords: Chordal graphs, Hamiltonian cycle, monotone polygon, parallel algorithms and complexity, visibility graphs.

1 Introduction

The Visibility Graph is an important combinatorial structure in computational geometry used in computing shortest paths amidst polygonal obstacles in the plane [7] and in decomposing 2D shapes into clusters [8]. The *visibility graph* G of a simple polygon P is a graph whose vertices $\{v_1, v_2, \ldots, v_n\}$ correspond to the vertices $\{p_1, p_2, \ldots, p_n\}$ of P and v_iv_j is an edge in G *iff* p_i is visible from p_j, *i.e.*, the line segment joining p_i and p_j does not intersect the exterior of P.

The general problem of reconstructing simple polygons from their visibility graphs still remains unsolved. Most results in this area have been

obtained by restricting the class of graphs or polygons, and also the developed algorithms have been sequential. El Gindy [3] provided an algorithm for reconstructing a representative embedding for any maximal outer-planar graph. Sreenivasa Kumar and Veni Madhavan [12] introduced the concept of 2-separator (in short, 2-sep) chordal graphs. The motivation behind introducing 2-sep chordal graphs comes from ElGindy's construction of polygons whose visibility graphs are maximal outer-planar graphs. The construction roughly proceeds as follows: Start with an equilateral triangle and fix one of its sides as a base line. Mount triangles on the other sides of the existing polygon such that appropriate visibility is maintained. This idea can be extended so that this type of construction can be done even if cliques of arbitrary size ≥ 3 take the place of triangles in a maximal outer-planar graph. Hamilonian 2-sep chordal graphs are precisely this kind of graphs. Therefore, the study of Hamiltonian 2-sep chordal graphs is important.

In [12], the authors showed that Hamiltonian 2-sep chordal graphs are visibility graphs of one-sided monotone polygons, by giving a linear time sequential algorithm to construct a one-side nonotone polygon for a given Hamiltonian 2-sep chordal graph.

So far only some special classes of graphs are shown to be visibility graphs. Researchers have shown these classes of graphs to be visibility graphs by proposing sequential algorithms to construct the appropriate visibility polygons. However, no parallel algorithms exist to construct visibility polygons for these known classes of visibility graphs. So, it would be interesting to design parallel algorithms for recognizing these known classes of visibility graphs and to construct visibility polygons for these known classes of visibility graphs in parallel.

In this paper, we first propose two characterizations of 2-sep chordal graphs. With the help of these characterizations, we present a parallel algorithm for recognizing 2-sep chordal graphs which runs in $O(\log^2 n)$ time using $O(mn)$ processors on the concurrent read concurrent write (CRCW), parallel random access machine (PRAM) model, where n is the number of vertices and m is the number of edges. We also present a new characterization of Hamiltonial 2-sep chordal graphs, based on which we propose parallel algorithms to recognize Hamiltonian 2-sep chordal graphs and to construct a Hamiltonian cycle for such a graph. The recognition of Hamiltonian 2-sep chordal graphs requires $O(\log^2 n)$ time and $O(mn)$ processors on the concurrent read exclusive write (CREW) PRAM model. Given a Hamiltonian 2-sep chordal graph, our parallel algorithm for constructing a Hamiltonian cycle takes $O(\log^2 n)$ time and $O(m)$ processors on the same CREW PRAM model. Finally, we propose a parallel algorithm for constructing one-sided Monotone polygons from Hamiltonian 2-sep chordal graphs. This algorithm requires $O(\log^2 n)$ time and uses $O(n)$ processors on the CREW PRAM model.

The rest of the paper is organized as follows. Section 2 introduces some preliminary concepts and summarizes results pertaining to our study. Section 3 presents the characterization and recognition of 2-sep chordal graphs. In Section 4, we propose a parallel algorithm for constructing one-sided monotone polygons from Hamiltonian 2-sep chordal graphs. The final section concludes the paper.

2 Preliminaries

Throughout this paper we use "iff" to stand for if and only if, "w.r.t" for with respect to, and "s.t." for such that. We consider only simple, finite and undirected graphs having n vertices and m edges.

A graph $G = (V, E)$ is called *chordal* if every cycle of length at least four, has a chord *i.e.*, an edge joining two non-consecutive vertices of the cycle. Let $G[S]$, $S \subseteq V$, denote the induced subgraph of G on S. If $G[C]$, $C \subseteq V$, is a complete subgraph of G, then C is called a *clique* of G. Let $N(v) = \{w \in V(G) \mid vw \in E(G)\}$, be the neighborhood (the set of adjacent vertices) of v. A vertex v is said to be *simplicial* if $G[N(v)]$ is a clique in G. A *perfect elimination ordering*(*peo*) $\sigma = (v_1, v_2, \ldots, v_n)$ of G is an ordering of the vertices of G s.t. v_i is simplicial in $G[\{v_i, v_{i+1}, \ldots, v_n\}]$, $1 \leq i \leq n$. A graph G is chordal iff it has a *peo* [5]. The *monotone adjacency set* of v_i w.r.t *peo* σ is defined as $N(v_i, \sigma) = \{v_j \in V(G) \mid j > i \text{ and } v_i v_j \in E(G)\}$.

A subset B of the vertex set of a chordal graph G is called a *base set* w.r.t a *peo* σ if $\exists$ a vertex v_i s.t. $B = N(v_i, \sigma)$ and B is not a maximal clique in $G[\{v_{i+1}, \ldots, v_n\}]$. The number of such vertices that exist for a particular base set is called its *multiplicity*, and is denoted by $\mu(B, \sigma)$.

The concept of base set was originally introduced in [11], and the following results on base sets have been established in [11] and [12]. We state them here for the sake of completeness.

Lemma 2.1 ([11]) *Let B be a base set of a chordal graph G w.r.t a peo σ. Then B is a clique and a vertex separator of G.*

Theorem 2.2 ([11]) *Let α and β be two peos of a chordal graph G. (i) If B is a base set w.r.t α then it is also a base set w.r.t β. (ii) $\mu(B, \alpha) = \mu(B, \beta)$*

Hereafter, no particular *peo* will be assumed while referring to base sets, unless it is specifically mentioned.

A chordal graph is called a *k-separator* (*k-sep*, in short) *chordal* graph [12] if every base set is of size k.

Lemma 2.3 ([12]) *A base set B of a k-sep chordal graph G is contained in exactly $\mu(B) + 1$ maximal cliques.*

Let $Q_1, Q_2, \ldots, Q_k$ be the maximal cliques of a Hamiltonian 2-sep chordal graph G. A tree T is associated with G, whose vertices $\{q_1, q_2, \ldots, q_k\}$ correspond to the maximal cliques of G and $q_i q_j$ is an edge in T iff Q_i and Q_j share a base set. T is called the *clique tree* of G.

The next theorem characterizes Hamiltonian 2-sep chordal graphs, and the corolary follows immediately.

Theorem 2.4 ([12]) *A 2-sep chordal graph G is Hamiltonian iff (i) every base set of G has multiplicity one, and (ii) for every maximal clique Q of G, $\exists$ a Hamiltonian cycle $C(Q)$ of Q s.t. each of the base sets contained in Q appears as an edge in $C(Q)$.*

Corollary 2.5 ([12]) *Every edge of the Hamiltonian cycle of a 2-sep chordal graph lies in exactly one maximal clique.*

The following lemma is equivalent to the second condition of Theorem 2.4, but is simple and easy to test.

Lemma 2.6 ([12]) *Let $Q = (V, E)$ be a complete graph on n vertices. Let S be a set of edges of Q, where $|S| \leq n$. There exists a Hamiltonian cycle $C(Q)$ for Q containing the edges in S iff the graph $Q' = (V, S)$ is either a cycle on n vertices or a disconnected graph in which each component is a singleton vertex* (i.e., *a trivial component*) *or a path.*

If $G - C$ is disconnected by a maximal separating clique C into components $H_i = (V_i, E_i)$, $1 \leq i \leq r$, $r \geq 2$, then $G_i = G[V_i \cup C]$, $1 \leq i \leq r$, is said to be a *separated subgraph* of G w.r.t C. Define $w(G_i) = \{v \in C \mid$ there is a vertex $w \in (V(G_i) \cup C)$ s.t. $vw \in E(G_i)\}$. Maximal cliques of G_i which contain $w(G_i)$ are called *principal cliques* of G_i. The existence of a principal clique in every separated subgraph of a chordal graphi is assured by this lemma.

Lemma 2.7 ([9]) *Every separated subgraph G_i of a chordal graph has a principal clique.*

A polygon P is said to be *monotone* w.r.t a line l if the intersection of any line perpendicular to l with P is a connected segment, possibly empty. P is called a *one-sided* monotone polygon w.r.t l, if l is parallel to one of the edges.

3 Characterization and Recognition of 2-Sep Chordal Graphs

The recognition of Hamiltonian 2-sep chordal graphs is performed in two steps: (1) testing if the given graph is a 2-sep chordal graph, and (2) checking if it has a Hamiltonian cycle. We present two characterizations of 2-sep chordal graphs, one of which is used to recognize them.

Let us characterize the base sets of a k-sep chordal graph G by showing that they are exactly all minimal separators of G.

Lemma 3.1 *Let G be a k-sep chordal graph. B is a base set of G iff it is a minimal separator.*

Proof: First we prove that every minimal separator of G is a base set. Let B ($|B| = k$) be a minimal separator of G and $H_1, H_2, \ldots, H_r$, $r \geq 2$, be the connected components of $G - B$. Let $G_i = G[V(H_i) \cup B]$, $2 \leq i \leq r$. Since B is a minimal separator, $\exists$ a vertex v of G_1 which is adjacent to all the vertices in B (see [5]). Let $(v_1, v_2, \ldots, v'_r)$ be a *peo* of G_1, with $v = v_{r'-k}$ and $\{v_{r'-k+1}, v_{r'-k+2}, \ldots, v'_r\} = B$, and let $(x_1, x_2, \ldots, x_n)$ be a *peo* of G with $x_i = v_i$ for $1 \leq i \leq r'$. Such *peo*s can be constructed [10]. Since B is a minimal separator, it is not a maximal clique in any of the components G_i and hence, it is not a maximal clique in G[$\{x_{r'-k+1}, x_{r'-k+2}, \cdots, x_n\}$]. Therefore, B is a base set w.r.t v.

Next we prove that all the base sets of G are minimal separators. Let B be a base set of G and not a minimal separator. By Lemma 2.1 B is a vertex separator. This means that a proper subset of B is a minimal separator of G. Therefore, by the sufficiency of Lemma 3.1, a proper subset of B with size less than k is a base set. This is a contradiction to the fact that all base sets are of size k in G. Hence, B is a minimal separator. □

We now present two characterizations of 2-sep chordal graphs, one of which can be used to construct a parallel algorithm for recognizing them.

Let G be a chordal graph. If G has one maximal clique, then G is a 2-sep chordal graph. If G has two maximal cliques, say Q_1 and Q_2, the G is a 2-sep chordal graph iff $|Q_1 \cap Q_2| = 2$. So, we consider chordal graphs with at least three maximal cliques.

Theorem 3.2 *The following three statements are equivalent for a chordal graph G having at least 3 maximal cliques.*

(1) G is a 2-sep chordal graph.

(2) For every maximal separating clique C of G, $|w(G_i)| = 2$ for all separated subgraphs G_i w.r.t C.

(3) G is biconnected and $|Q_1 \cap Q_2| \leq 2$, where Q_1 and Q_2 are any two distinct maximal cliques of G.

Proof: Let us prove that $1 \Rightarrow 2$. Let C be a maximal clique of G and let G_i, $i \geq 2$, be the separated subgraphs of G w.r.t C. As C is a vertex separator of G, by its definition $w(G_i)$ is a vertex separator of G for all i, $1 \leq i \leq r$. Again, by Lemma 2.7, G_i has a principal clique, implying $\exists$ a vertex $x_i \in G_i - w(G_i)$ s.t. x_i is adjacent to all the vertices of $w(G_i)$. Also, since C is a maximal clique, $\exists$ at least one vertex in $C - w(G_i)$. So, $w(G_i)$ is a minimal vertex separator. Hence by Lemma 3.1, $w(G_i)$ is a base set of G for all i, $1 \leq i \leq r$. Since G is a 2-sep chordal graph, $|w(G_i)| = 2$ for all i, $1 \leq i \leq r$.

Next we prove that $2 \Rightarrow 3$. If possible, let v be a cut vertex of G. Let $H_1, H_2, \ldots, H_r$, where $r \geq 2$, be the connected components of $G - \{v\}$. Let $G_i = G[V(H_i \cup \{v\})]$, $1 \leq i \leq r$. Let C be a maximal clique of G_1 containing v. Now, consider the separated subgraphs of G w.r.t C. Clearly, each G_i, $2 \leq i \leq r$, $r \geq 2$, is a separated subgraph of G w.r.t C. Now, $|w(G_i)| = 1$ for $2 \leq i \leq r$ which is a contradiction to statement (2) of the theorem. In other words, G is biconnected.

If possible, let Q_i and Q_j be two maximal cliques of G s.t. $|Q_i \cap Q_j| \geq 3$. Since G has at least 3 maximal cliques, either Q_i or Q_j is a separating clique of G. Without loss of generality, let Q_i be a separating clique and G_i, $i \geq 2$, be the separated subgraphs of G w.r.t Q_i. Again, without loss of generality, assume that G_1 contains Q_j. Since $|Q_i \cap Q_j| \geq 3$, we get $|w(G_1)| \geq 3$ which is a contradiction to statement (2). Hence statement (3) is true.

Finally we prove that $3 \Rightarrow 1$. Consider any base set B. It follows from the definition of a base set that B lies in at least 2 maximal cliques (say Q_i and Q_j). Therefore, $|B| \leq |Q_i \cap Q_j|$. But, $|Q_i \cap Q_j| \leq 2$, which implies that $|B| \leq 2$. Since G is biconnected, any separator should be of size greater than 1. So $|B| = 2$ and hence, G is a 2-sep chordal graph. □

Statement (3) of Theorem 3.2 leads to a parallel 2-sep chordal graph recognition algorithm described below. For this algorithm, we assume the CRCW PRAM model of computation.

Algorithm* 2-*Sep_Chordal

$flag = TRUE$

Step 1. Test, in parallel, whether G is biconnected. If not, set $flag = FALSE$ and go to Step 9.

Step 2. Check, in parallel, if G is chordal. If not, set $flag = FALSE$ and go to Step 9.

Step 3. Find the maximal cliques of G.
/* Let $Q_1, Q_2, \ldots, Q_r$ be the maximal cliques of G */

Step 4. If $r = 1$, go to Step 9.

Step 5. If $r = 2$, check if $|Q_1 \cap Q_2| = 2$. If no, set $flag = FALSE$ and go to Step 9. (Note that $Q_1 \cap Q_2$ is the set of vertices having degree $n-1$.)

Step 6. Form a list L consisting of triplets (Q_i, Q_j, v) with $i < j$ and v contained in both Q_i and Q_j.

Step 7. Lexicographically sort L on the first two components.

Step 8. Check, in parallel, if three consecutive triplets of L have the same first and second components. If yes for some processor, set $flag = FALSE$.

Step 9. If $flag$ is $TRUE$, then G is 2-sep chordal, else it is not.

Because of statement (3) of Theorem 3.2, the above algorithm correctly tests if a given graph is a 2-sep chordal graph. Below we show its complexity.

Theorem 3.3 *A 2-sep chordal graph can be recognized in $O(\log^2 n)$ time using $O(mn)$ processors on a CRCW PRAM model.*

Proof: The biconnectivity (Step 1) can be tested in $O(\log^2 n)$ time using $O(n^2/\log^2 n)$ processors [13]. Checking if G is chordal and finding its maximal cliques (Steps 2-3) requires $O(\log^2 n)$ time using $O(m+n)$ processors [6]. Step 5 takes $O(\log n)$ time using $O(m+n)$ processors . Let n_v be the number of maximal cliques containing v. Then the size of the list L is $\sum_{v \in V(G)} \binom{n_v}{2} \leq \sum_{v \in V(G)} n_v^2 \leq A_1 n \sum_{v \in V(G)} n_v \leq A_2 nm$, for some constants A_1 and A_2. Therefore, the number of triplets in L is $O(mn)$. Step 7 can be performed in $O(\log mn)$ $(= O(\log n^3) = O(\log n)$, as $m = O(n^2))$ time with $O(mn)$ processors using a parallel merge sort algorithm [2]. Step 8 takes constant time using $O(mn)$ processors. Hence the theorem. □

3.1 Hamiltonian 2-sep chordal graphs

Let us now discuss the problem of recognizing whether a given 2-sep chordal graph is Hamiltonian. For the following two algorithms we assume the CREW PRAM model of computation.

Note that, a graph G is a set of cycles or paths iff the maximum degree of the graph is at most three. This fact, together with Theorem 2.4 and Lemma 2.6, gives rise to the following alternate characterization of Hamiltonian 2-sep chordal graphs.

Theorem 3.4 *A 2-sep chordal graph G is Hamiltonian iff*

(1) Every edge of G is contained in exactly two maximal cliques.

(2) No three base sets lying in the same maximal clique have a vertex in common.

(3) For every maximal clique Q of G, either Q' is connected or each non-trivial connected component of Q' has exactly two vertices of degree 1, where $Q' = (Q, S)$ and $S = \{(u, v) \mid \{u, v\}$ is a base set of G contained in $Q\}$.

The following parallel recognition algorithm for Hamiltonian 2-sep chordal graphs is based on the above characterization.

Algorithm Ham_2-Sep_Chordal

$flag = TRUE$

Step 1. /* This step checks if each edge is contained in exactly two maximal cliques. This is done by forming a list of pairs of edges with the maximal cliques that contain them and by checking if there are exactly two pairs with the same edge after sorting this list. If this is not true then the first condition in the above theorem fails and hence, G is not a Hamiltonian 2-sep chordal graph. */

1.1 For every maximal clique Q_i of G, insert pairs (e_j, Q_i) into a list L^1 for each edge e_j in Q_i.

1.2 Lexicographically sort L^1 on the first coordinate.

1.3 In parallel, check if three consecutive pairs have the same first coordinate. If yes for some processor, set $flag = FALSE$ and go to Step 4.

Step 2. /* This step checks if the second condition of Theorem 3.4 holds. This is done by forming lists L_i^2 of base sets corressponding to each maximal clique and by checking if any vertex appears thrice in one of these lists after sorting them in parallel*/

2.1 For every maximal clique Q_i, perform Steps 2.2 to 2.4 in parallel.

2.2 For every base set $B = \{a, b\}$ in Q_i, insert the ordered pair (a,b) into a list L_i^2 corresponding to the maximal clique.

2.3 Sort the lists L_i^2 on the first coordinate.

2.4 In parallel, check if any three consecutive pairs have the same first coordinate. If yes for some processor, set $flag = FALSE$ and go to Step 4.

Step 3. For each maximal clique Q_i do in parallel:
/* Let $|Q_i| = k$ and $Q_i' = (V', E')$, where $V' = Q_i$ and $E' = \{(u, v) \mid \{u, v\}$ is a base set contained in $Q_i\}$ */

3.1 Find all the connected components of Q_i'.

3.2 If Q_i' is connected (i.e., it has exactly one component), then check if it has either zero or two degree-1 vertices. If no, set $flag = FALSE$ and go to Step 4.

3.3 For each vertex v of Q_i', insert the non-trivial connected component of Q_i' containing v, say $\mathcal{A}$, into a list L_i^3 if v is a degree-1 vertex in $\mathcal{A}$.

3.4 Sort the list L_i^3.

3.5 In parallel, check if there are exactly two entries for each non-trivial connected component. If no for some processor, set $flag = FALSE$.

Step 4. If $flag = TRUE$ then G is Hamiltonian.

The proof of correctness of this algorithm follows from Theorem 3.4. Note that Step 1 also generates the base sets of the graph G. These are given by those edges e_j that lie in exactly two maximal cliques.

Theorem 3.5 *A 2-sep chordal graph can be tested for Hamiltonicity in* $O(\log^2 n)$ *time using* $O(mn)$ *processors on a CREW PRAM model.*

Proof: The number of pairs in the list L^1 is $O(mn)$ because there are at most n maximal cliques. The number of pairs in L_i^2 is $O(m)$ and the number of cliques is $O(n)$. Therefore, there is a total of $O(mn)$ pairs in all the L_i^2s put together. The list L_i^3 has $O(k)$ elements and so, the total number of entries in all the L_i^3s put together is $O(\sum |Q_i|) = O(m)$. So, Steps 1.2, 2.3 and 3.4 can be implemented in $O(\log mn)$ $(= O(\log n^3) = O(\log n)$, as $m = O(n^2))$time with $O(mn)$ processors using a parallel merge sort algorithm [2]. Steps 1.3, 2.4, 3.2 and 3.5 take constant time. The connected components of Q_i' $(|Q_i'| = k)$ can be found in $O(\log^2 k)$ time using $O(k^2/\log^2 k)$ processors [1]. By a similar analysis as for the number of entries in L_i^3, the number of processors required to find the connected components of Q_i' for all i, $1 \leq i \leq r$ is $O(mn)$. Hence the theorem. □

Next we propose a parallel algorithm to construct a Hamiltonian cycle in a Hamiltonian 2-sep chordal graph.

Algorithm Create_Ham_Cycle

Step 1. For each ordered pair (v_j, Q_i), where v_j is a vertex in the maximal clique Q_i, do Steps 2 to 4 in parallel.

Step 2. If v_j is a degree-1 vertex in Q_i' as defined in the previous algorithm, insert the pair $(\mathcal{A}_{v_j}, v_j)$ to a list L_i, where $\mathcal{A}_{v_j}$ is the connected component of Q_i' in which v_j lies.

Step 3. If v_j is an isolated vertex in Q_i', insert $(0, v_j)$ into the list L_i.

Step 4. Sort L_i on the first coordinate.

Step 5. For every pair of consecutive elements $(\mathcal{A}_{v_a}, v_a)$ and $(\mathcal{A}_{v_b}, v_b)$ of L_i, insert the edge (v_a, v_b) into the list $C(Q_i)$ if they differ in the first coordinate or both have '0' as the first coordinate. (Note $C(Q_i)$ is the Hamiltonial cycle of the clique Q_i.)

Step 6. For each of these lists L_i, insert edge (v_f, v_l) into $C(Q_i)$, where v_f is the second coordinate of the first entry and v_l is the second coordinate of the last entry in L_i.

Step 7. Take the union of all lists $C(Q_i)$ to result in $C(G)$, the Hamiltonian cycle of G.

The complexity of the above algorithm is proved below.

Theorem 3.6 *A Hamiltonian cycle can be constructed in a 2-sep chordal graph in $O(\log^2 n)$ time using $O(m)$ processors on a CREW PRAM model.*

Proof: Let G be a Hamiltonian 2-sep chordal graph $S(G)$ be the set of all maximal cliques of G. Then, $\bigcup_{Q \in S(G)} C(Q) - B(Q)$ is a Hamiltonian cycle of G, where $C(Q)$ is a Hamiltonian cycle of the clique Q containing all the base sets in Q and $B(Q)$ is the set of all edges in Q which correspond to base sets in G. Therefore, algorithm Create_Ham_Cycle correctly constructs a Hamiltonian cycle of the given Hamiltonian 2-sep chordal graph. The number of pairs in all the L_is put together is $O(m)$. Step 4 of the algorithm requires $O(\log^2 n)$ time using $O(m)$ processors. The Hamiltonian cycle of G, *i.e.*, $C(G)$ is obtained as a set of $n-1$ edges. Given this edge list, using the Euler tour technique [4], we can order the vertices along the Hamiltonian cycle in $O(\log^2 n)$ time using $O(n)$ processors. Thus, the theorem is proved. □

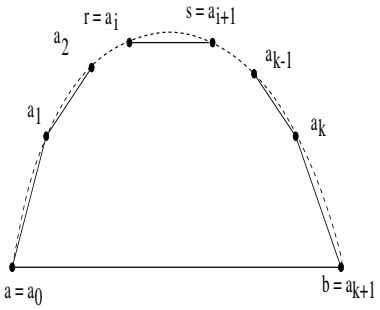

Figure 1: The convex polygon corresponding to Q_1

4 Polygon Reconstruction

In this section we present a parallel algorithm for constructing a one-sided monotone polygon (w.r.t the X-axis) whose visibility graph is a Hamiltonian 2-sep chordal graph. Before proceeding further, let us give a brief overview of the sequential algorithm for this construction due to Sreenivasa Kumar and Veni Madhavan [12].

Let ab be an edge in $C(G)$, the Hamiltonian cycle of G. By Corollary 2.5, ab is contained in a unique maximal clique, say Q_1. Draw the convex polygon corresponding to Q_1 with edge ab parallel to the X-axis as shown in Figure. 1. Note that ab is the base line for Q_1. To embed the remaining vertices, the clique tree T of G that is rooted at Q_1, is traversed in a depth-first-search (DFS) manner and the convex polygons corresponding to the maximal cliques are constructed, with the base line given by the base set B_i that corresponds to the edge joining Q_i and its parent. Two points p_1 and q_1, are determined taking the visibility constraints into consideration. The vertices of $Q_i - B_i$ are plotted on an arc passing through p_1 and q_1 and between the vertical lines at p_1 and q_1, the order in which they appear being given by $C(Q_i)$. The point p_1 is calculated as the intersection of the line passing through b and s with the vertical line through r (see Figure 2). Similarly, q_1 is calculated as the point of intersection of the line passing through a and r with the vertical line through s. The two special cases, (i) $r = a$ and $s = a_1$ and (ii) $r = a_{k-1}$ and $s = b$ are handled separately [12].

The proposed parallel algorithm for the polygon construction is described below. For this purpose, we need the CREW PRAM model of computation. Since the DFS traversal of the clique tree leads to a linear time algorithm, we avoid its use in the following manner. Given the clique tree T as an edge list, we calculate the shortest path of each vertex in T from the root of T. Note that we consider Q_1 as the root of T as mentioned in the previous paragraph. This gives the depth of each maximal clique in T. We then construct in parallel the convex polygons corresponding to the maximal cliques. We fix the x-coordinate of the vertices of the polygon by setting them equal to their label, *i.e.*, $x_i = i$. Therefore, the construction is essentially the calculation of the y-coordinate for each vertex.

Suppose we are to draw the convex polygon corresponding to the maximal clique Q_i, with the base line as the base set B_i and whose depth in

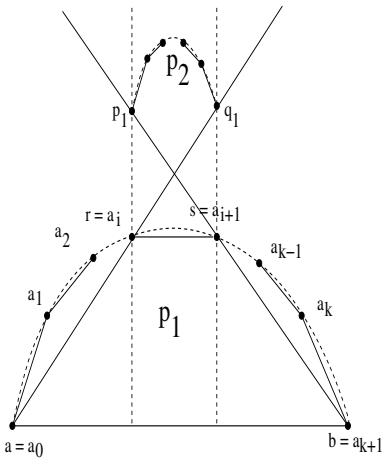

Figure 2: Constructing the convex polygon with base line (r, s)

T is t. Once the points p_1 and q_1 are calculated, we set the y-coordinates of p_1 and q_1 to $y'_{p_1} = max(y_{p_1}, y_{q_1})$ and plot all the vertices of $Q_i - B_i$ on the parabola passing through $p_1(x_{p_1}, y_{p_1})$ and $q_1(x_{p_1}, y_{p_1})$. This parabola is given by the equation

$$y = y'_{p_1} + (x_s - x)(x - x_r) \tag{1}$$

where $B_i = \{r, s\}$.

Lemma 4.1 *If p_1 and q_1 are the points as defined in Figure 2, then the y-coordinate of p_1 is given by*

$$y_t = \left\lceil \left(\frac{D' - k}{n-2} \right) \left((n-2)^t - 1 \right) + kt^2 \right\rceil \tag{2}$$

$$where\ k = \frac{n^2}{4(2t-1)(1-n)} \tag{3}$$

$$and\ D' = \left\lfloor \left(\frac{n-1}{2} \right)^2 \right\rfloor$$

Proof: Just as p_1 corresponds to the base of polygon P_2, let p_2 and p_3 correspond to the base of the polygons that are constructed in the previous two steps before P_2, i.e. p_2 corresponds to the base of polygon P_1 in Figure 2. The point p_1 is calculated as the intersection of the line passing through s and b with the vertical line at r. So,

$$\frac{y_{p_1} - y_b}{r - b} = \frac{y_{p_2} + c_s - y_b}{s - b}$$

where p_2 is the point corresponding to the polygon P_1 (see Figure 2) and c_s is the vertical displacement of s from p_2. Hence,

$$y_{p_1} = \left(\frac{r-b}{s-b} \right) (y_{p_2} + c_s - y_b) + y_b.$$

Now,

$$\left(\frac{r-b}{s-b} \right) \leq n - 1$$

and

$$c_s \leq \left(\frac{x_s - x_r}{2}\right)^2 \leq \left\lfloor \left(\frac{n-1}{2}\right)^2 \right\rfloor .$$

From Figure 2, it is clear that p_1 can have y-coordinate greater than y_{p_1} and still satisfy the visibility criteria. So, we substitute these upper bounds in the expression for y_{p_1} to get

$$y_{p_1} = (n-1)(y_{p_2} + \left\lfloor \left(\frac{n-1}{2}\right)^2 \right\rfloor - y_b) + y_b.$$

Expressing y_b in terms of the displacement of b from the point p_3, which was used to plot a and b, we have

$$y_{p_1} = (n-1)\left(y_{p_2} + \left\lfloor \left(\frac{n-1}{2}\right)^2 \right\rfloor\right) + (2-n)(y_{p_3} + c_b).$$

Again, $|c_b| \leq n-1$. Substituting this bound, rearranging the terms and setting the y-coordinate of p_1 to this bound we get

$$y_{p_1} = (n-1)y_{p_2} + (2-n)y_{p_3} + \left\lfloor \left(\frac{n-1}{2}\right)^2 \right\rfloor .$$

This is a recurrence relation in the depth t of the maximal clique in the clique tree T and can be rewritten as

$$y_t = (n-1)y_{t-1} + (2-n)y_{t-2} + \left\lfloor \left(\frac{n-1}{2}\right)^2 \right\rfloor$$

where y_t is the y-coordinate of the point m_i corresponding to the maximal clique at depth t in T. Solving this recurrence we get

$$y_t = \underbrace{\alpha(n-2)^t + \beta}_{general\ solution} + \underbrace{kt^2}_{particular\ solution}$$

where k is given by Eq. 3.

The initial conditions are $y_0 = 0$ and $y_1 = D$ (say). The upper bound for D is $\lfloor (n-1/2)^2 \rfloor$ which we denote by D'. Substituting this value and taking the ceiling of the resultant we get an integral value for y_t which is given by Eq. 2. □

The coordinate y_{q_1} can be calculated similarly. The following algorithm gives the entire construction.

Algorithm Construct_Polygon

Step 1. Let the Hamiltonian cycle of G be $C(G) = v_1, v_2, \ldots, v_n$ and the maximal cliques be $Q_1, Q_2 \ldots Q_k$.

Step 2. Choose an edge $v_i v_j$ in $C(G)$ and draw the line segment joining the points$(1,0)$ and $(n,0)$. This is the base line of G.

Step 3. Relabel the vertices from v_i to v_j along the Hamiltonian cycle as $1, 2, \ldots, n$.

Step 4. Set Q_1, the maximal clique containing $v_i v_j$, as the root of the clique tree T.

Step 5. For $i = 1$ to n, in parallel, find y_{p_i} for the point p_i corresponding to Q_i using Eq. 2 (The depth of Q_i in T is calculated as the length of the shortest path from the root to Q_i).

Step 6. For $v = 1$ to k, in parallel, plot the point v of the clique Q_i as $(v, y_p + (x_s - v)(v - x_r))$ where (r, s) is the base line of the clique.

Step 7. For $i = 1$ to n, in parallel, draw the line segment joining the points i and $i + 1$.

Theorem 4.2 *A one-sided monotone polygon, whose visibility graph is a Hamiltonian 2-sep chordal graph, can be reconstructed, given the graph along with a Hamiltonian cycle, in $O(\log^2 n)$ time using $O(n)$ processors on a CREW PRAM model.*

Proof: Step 3 requires $O(\log^2 n)$ time using $O(n)$ processors. Given the clique tree T, in the form of an edge list, it can be rooted to get the parent-child information in $O(\log n)$ time using $O(n)$ processors [13]. Once the parent-child information is available we can find the depth of each node using the standard doubling technique. So, Step 5 takes $O(\log^2 n)$ time using $O(n)$ processors. All other steps require constant time. □We make the following remark. One important difference in the polygons constructed by our algorithm and the approach in [12] is that all the vertices of the polygon constructed by our algorithm have integer coordinates. This is ensured by the fact that (i) the x-coordinates are fixed by setting them to the value of the vertex labels and (ii) the vertices are plotted on the parabola $y = y'_{p_1} + (x_s - x)(x - x_r)$. So, the y-coordinates are integers for integral values of x. Due to this feature, our algorithm is practical and involves lesser computation than that required for floating point operations as in [12].

5 Conclusions

We have presented a parallel algorithm to recognize Hamiltonian 2-sep chordal graphs that run in $O(\log^2 n)$ time using $O(mn)$ processors on a CRCW PRAM model. We have also given a parallel algorithm to construct one-sided monotone polygons from Hamiltonian 2-sep chordal graphs which has a running time of $O(\log^2 n)$ time and uses $O(n)$ processors on a CREW PRAM model, once the Hamiltonian cycle is given. The Hamiltonian cycle construction takes $O(\log^2 n)$ time using $O(m)$ processors on a CREW PRAM model.

Acknowledgment

The authors are grateful to the anonymous referees for their helpful suggestions and constructive comments which greatly improved the presentation of the paper.

References

[1] F.Y. Chin, J. Lam, and I. Chen. Efficient parallel algorithms for some graph problems. *Communications of ACM*, 25(9):659–665, 1982.

[2] R. Cole. Parallel Merge Sort. *SIAM J. Computing*, 17:770–785, 1988.

[3] H. El Gindy. *Hierarchial decomposition of polygons with applications.* PhD thesis, McGill University, Montreal, Quebec, 1985.

[4] A. Gibbons and W. Rytter. *Efficent parallel algorithms.* Cambridge University Press, 1988.

[5] M.C. Golumbic. *Algorithmic graph theory and Perfect Graphs.* Academic Press, 1980.

[6] P. N. Klein. Efficient Parallel Algorithms for Chordal graphs. *SIAM J. Computing*, 25(4):797–827, August 1996.

[7] T. Lazano-Perez and M.A. Wesley. An algorithm for planning collision free paths among polyhedral obstacles. *Communications of ACM*, 22:560–570, 1979.

[8] Shapiro L.G. and R.M. Haralick. Decomposition of two-dimensional shape by graph-theoretic clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1:10–19, 1979.

[9] B.S. Panda and S.P. Mohanty. Intersection graphs of vertex disjoint paths in a tree. *Discrete Mathematics*, 146:179–209, 1995.

[10] B.S. Panda, S.P. Mohanty, and S.B. Rao. Chordal Graphs with specified Perfect Elimination Orderings. *Journal of Ramanujan Mathematical Society*, 13(2):61–72, 1998.

[11] P. Sreenivasa Kumar and C.E.Veni Madhavan. A New Class of Separators and Planarity of Chordal Graphs. In *Proc. Ninth FST&TCS Conference, LNCS*, volume 405, pages 30–43, 1989.

[12] P. Sreenivasa Kumar and C.E.Veni Madhavan. 2-Separator Chordal Graphs. In *Proc. Third National Conference on Theoritical Computer Science, IIT Kharagpur*, pages 37–52, 1993.

[13] R.E. Tarjan and U. Vizhkin. An efficient parallel Biconnectivity algorithm. *SIAM Journal of Computing*, 14:862–874, 1985.