1998 International Computer Symposium
Workshop on Algorithms
December 17-19, 1998, N.C.K.U., Tainan, Taiwan, R.O.C.

Parallel Algorithms for the Hamiltonian Circuit Problem in Convex Bipartite Graphs and Circular Convex Bipartite Graphs

Y. Daniel Liang
Department of Computer Science
Indiana Purdue University at Fort Wayne
Fort Wayne, IN 46805
email: liangy@ipfw.edu

Maw-Shang Chang
Department of Computer Science and Information Engineering
National Chung Cheng University
Min-Hsiung, Chiayi 621, Taiwan
Republic of China
email: mschang@cs.ccu.edu.tw

Abstract

The problem of deciding whether a graph G has a Hamiltonian circuit is $\mathcal{NP}$-complete. The problem remains $\mathcal{NP}$-complete even if G is bipartite. Polynomial algorithms for the Hamiltonian circuit problem are found in cocomparability graphs, circular-arc graphs, convex bipartite graphs and circular convex bipartite graphs. However, no $\mathcal{NC}$ algorithms for this problem have been found on any graphs except on trivial cases such as the graphs of degree 2 or less. This paper presents an $O(\log^2 n)$ time, n^2 processor EREW PRAM algorithm to determine if there is a Hamiltonian circuit in a convex bipartite graph. If there is a Hamiltonian circuit, we can find it within the same resource bounds. This paper also presents an $O(\log^2 n)$ time, n processor EREW PRAM algorithm to determine and find a Hamiltonian circuit in a circular convex bipartite graph, assume that the input graph is not a convex bipartite graph.

Keywords: Circular Convex Bipartite Graphs, Convex Bipartite Graphs, Hamiltonian Circuits, $\mathcal{NC}$, Maximum Matching, Parallel Algorithms.

1 Introduction

The *Hamiltonian circuit* (HC) problem asks whether there is a cycle in a graph that passes through each node exactly once. If such a cycle exists, it is called a Hamiltonian circuit of the graph. The importance of this problem is well-known and widely documented in the literature. Below we mention a few of the results regarding HC. Throughout this paper we will be concerned only with undirected graphs.

The problem of deciding whether a graph G has an HC is $\mathcal{NP}$-complete [14]. The problem remains $\mathcal{NP}$-complete even if G is bipartite or planar, cubic, 3-connected, and has no face with fewer than five edges [6, page 199]. If either the starting point, ending point, or both are specified, the problem is still $\mathcal{NP}$-complete [6, page 200]. The problem can be solved in polynomial time if G has no vertex with degree greater than two or if G is a line graph [6, page 199]. In the former case the problem is easily seen to be in $\mathcal{NC}^2$. The problem is polynomially solvable in cocomparability graphs, a superclass of many special graphs such as trapezoid graphs, permutation graphs and interval graphs. The problem can also be solved in time $O(n^2 \log n)$ if G is a circular-arc graph (a generalization of interval graph), where n is the number of vertices in G [17]. If the circular-arc model is given with presorted endpoints, the problem can be solved in $O(n)$ time [13]. The HC problem in a circular convex bipartite graph (a generalization of convex bipartite graph) was recently solved in linear time by reducing in linear time to the same problem in an appropriate circular-arc model [12].

$\mathcal{NC}$ algorithms have been developed on chordal graphs, a superclass of interval graphs, for finding the following: all maximal cliques, an intersection graph representation, an optimal coloring, a perfect elimination scheme, a maximum independent set and a minimum clique cover [15]. Several of these results were improved in [9]. In [1], weighted versions of several of these and related problems are shown to be in $\mathcal{NC}$ for circular-arc graphs. It is not difficult to see that the sequential algorithms in [2] for the domination problem and its variants (connected, total, independent domination) in convex bipartite graphs can be parallelized and extented to solve the same problems in circular convex bipartite graphs. However, no $\mathcal{NC}$ algorithms have been found for the HC problem except in

some trivial cases such as the graphs of degree 2 or less.

In this paper we present an $O(\log^2 n)$ time, n^2 processor EREW PRAM algorithm to determine if there is a Hamiltonian circuit in a convex bipartite graph. If there is a Hamiltonian circuit, we can find it within the same resource bounds. The result is obtained by exhibiting two subpaths and the concatenation of these two subpaths results in an HC. We also present an $O(\log^2 n)$ time, n processor EREW PRAM algorithm to determine and find a Hamiltonian circuit in a circular convex bipartite graph, assume that the input graph is not a convex bipartite graph. The algorithm is supported by the fact that if G is a circular convex bipartite graph not a convex bipartite graph and $G = (A, B, E)$ has an HC, then there exists an HC in G that connects every two neighboring vertices in A using vertices in B.

The remainder of this paper is outlined as follows. In section 2, we describe some background material and several results necessary for obtaining our algorithms. In section 3, we introduce a parallel algorithm for finding a special complete path cover on A in a CBG $G = (A, B, E)$. This algorithm is a preliminary result for establishing the main algorithms. In section 4 and 5, we give the parallel algorithms for finding an HC in convex bipartite graphs and in circular convex bipartite graphs, respectively.

2 Preliminaries

We assume the reader is familiar with basic complexity theory, the complexity class $\mathcal{NC}$, and the EREW PRAM model of computation. For background material in these areas, see for example one of the following: [5, 8, 10]. Let n denote a problem's input size. As is customary we call an *$\mathcal{NC}$ algorithm*, if the algorithm that runs on a PRAM in $\log^k n$ time for some constant k while using only a polynomial number of processors.

A common approach in complexity theory for studying problems that are $\mathcal{NP}$-complete on arbitrary graphs is to restrict the problems to a large subclass of graphs. Convex bipartite graphs and circular convex bipartite graphs are an important class of graphs that we focus our attention on. They arise in resource allocation problems and in a number of scheduling problems.

Let $G = (A, B, E)$ be a bipartite graph with two distinct sets of vertices $A = \{a_1, \cdots, a_n\}$ and $B = \{b_1, \cdots, b_m\}$. A *bipartite graph* G is *convex* if for any $b \in B$ and $a_i, a_j \in A$ with $i < j$, $(a_i, b) \in E$ and $(a_j, b) \in E$ implies that $(a_u, b) \in E$, for $i \leq u \leq j$. In other words, a bipartite graph $G = (A, B, E)$ is convex if there is a total ordering on A such that for any $b \in B$ the set of vertices of A connected to b forms an interval in this ordering. A *bipartite graph* $G = (A, B, E)$ is *circular convex* if for any $b \in B$ and $a_i, a_j \in A$ with $a_i < a_j$ such that $(a_i, b) \in E$ and $(a_j, b) \in E$ implies either $(a_i, b) \in E$, $(a_{i+1}, b) \in E$, $\cdots$, $(a_j, b) \in E$ or $(a_j, b) \in E$, $(a_{j+1}, b) \in E$, $\cdots$, $(a_n, b) \in E$, $(a_1, b) \in E$, $\cdots$, $(a_i, b) \in E$. In other words, a bipartite graph $G = (A, B, E)$ is circular convex if there is a circular ordering on A such that for any $b \in B$ the set of vertices of A connected to b forms a circular arc in this ordering.

Assume that the convex bipartite graph (CBG) $G = (A, B, E)$ is given by specifying the total ordering on $A = \{a_1, \cdots, a_n\}$ with $a_1 < \cdots < a_n$. For every vertex $b \in B$ in G, the *left-end neighbor* and *right-end neighbor* of vertex b , denoted by $BEG[b, G]$ and $END[b, G]$, are the smallest and largest vertices, respectively, of A connected to b in G. (G may be omitted in the notation for simplicity if G is clearly noted in the context.) Let $B = \{b_1, \cdots, b_m\}$ with $END[b_i] < END[b_j]$ or $END[b_i] = END[b_j]$ and $BEG[b_j] \leq BEG[b_i]$ for $1 \leq i < j \leq m$. We call a CBG satisfying this ordering to be in *compact form.*

Assume that the CCBG $G = (A, B, E)$ is given by specifying the circular ordering on $A = \{a_1, \cdots, a_n\}$ in a circle with $a_1 < \cdots < a_n$. For every $b \in B$ in G, let $BEG[b, G]$ and $END[b, G]$ denote the clockwise first and last elements, respectively, of the consecutive elements of A connected to b in G.

Our algorithms make use of the maximum matching problems in CBG and CCBG. Glover [7] gave a greedy algorithm for finding a maximum matching M in a CBG G. His algorithm is an implementation of the following simple idea. For each a_i, $i = 1, \cdots, n$, if a_i is adjacent to at least one unmatched vertex in B, select an unmatched smallest vertex $b \in B$ adjacent to a_i and add (a_i, b) to M. Otherwise, simply leave a_i unmatched. Dekel and Sahni [3] gave a parallel algorithm for the maximum matching problem using the binary tree method of [4]. Their result is summarized in the following lemma.

Lemma 2.1 *The maximum matching problem in convex bipartite graphs can be solved on an EREW PRAM in time $O(\log^2 n)$ using n processors. Furthermore, the parallel maximum matching algorithm produces the same matching by Glover's algorithm.*

We now describe a parallel algorithm for finding a MM in a CCBG. Liang and Blum [12] gave a linear time algorithm for finding a maximum matching in a CCBG by applying Glover's algorithm twice on subgraphs G' and G'' of a CCBG. G' is defined directly from the structure of G as follows.

$BEG[b, G'] = BEG[b, G]$

$END[b, G'] = END[b, G]$, if $BEG[b, G] \leq END[b, G]$;

$END[b, G'] = n + END[b, G]$, if b is a boundary vertex;

Note that if $BEG[b, G] > END[b, G]$, b is adjacent to a_1 and a_n. Such a vertex b is called a *boundary vertex.* After obtaining a maximum matching M' in G' using Dekel and Sahni's parallel algorithm, G'' can be defined as follow: for each unmatched boundary vertex $b \in B$, let $BEG[b, G''] = a_1$ and $END[b, G''] = END[b, G]$, and for each other vertex $b \in B$, let $BEG[b, G''] = BEG[b, G]$ and $END[b, G''] = END[b, G']$. It was shown in [12] that a maximum matching M'' in G'' is also a maximum matching in G. Thus, finally applying Dekel and Sahni's parallel algorithm on G'' we obtain a MM in G. Hence, we have the following result.

Lemma 2.2 *The maximum matching problem in circular convex bipartite graphs can be solved on an EREW PRAM in time $O(\log^2 n)$ using n processors. Furthermore, the parallel maximum matching algorithm produces the same matching by Liang and Blum's algorithm.*

3 Find a complete path on A in a CBG

A *complete path on* A is a path that connects all vertices in A using $|A|-1$ vertices in B. We first give the following algorithm to find a complete path in G. This algorithm is a stepping stone to establish the HC algorithm in a CBG.

Algorithm: Greedy-Path-CBG (Finding a complete path in a CBG)
Input: A CBG $G = (A, B, E)$ in compact form.
Ouput: A complete path on A, or a message stating that G has no complete path.

1 $P \leftarrow \{a_1\}$;
2 **for** $i \leftarrow 2, \cdots, n$ **do**
3 **if** there exists an unmatched vertex $b \in B$ such that b is adjacent to a_{i-1} and a_i **then**
4 Select the smallest such vertex b, remove b from B, and append b, a_i into P;
5 **else** report no complete path, exit
6 **endfor**.

Lemma 3.1 *Algorithm Greedy-Path-CBG returns a complete path iff G has a complete path.*

Proof. Suppose that the algorithm terminates at $i = k < n$. Let the partial path be $P = a_1, b_{i_1}, a_2, b_{i_2}, \cdots, a_{k-1}$. Two sets are *disjoint* if no vertices in the two sets are adjacent. We begin the following procedure to remove some h number of vertices from B and show that the remaining graph has $h+2$ pair-wise disjoint sets where each set contains at least one vertex in A.
Step 1. Let $F_1 = \{a_k, a_{k+1}, \ldots, a_n\}$ $\cup\{b : b \in B, BEG[b,G] \geq a_k\}$ and $B' = \{b : b \in B \setminus (F_1 \cup P)\}$. Clearly, $END[b,G] < a_k$ for every $b \in B'$. Let $R \subset B$ be the set of vertices in P adjacent to a_k. It is easy to see that every vertex in F_1 is not adjacent to any vertex in $P \setminus R$. It is also easy to see that every vertex in B' is not adjacent to any vertex in F_1. Suppose R is empty. Clearly F_1 and $P \cup B'$ are two disjoint sets. In other words, G is not connected. Thus, the procedure stops if R is empty.
Step 2. Let r be the maximum vertex in R. Removing r from G, P is divided into two paths P_1 and P_2 where $P = P_1 \rightarrow r \rightarrow P_2$. By Algorithm Greedy-Path-CBG , no vertex of B in P_2 is adjacent to any vertex of A in P_1 and every vertex in F_1 is not adjacent to any vertex in $P_1 \setminus R$ and $P_2 \setminus R$. By similar arguments, if a vertex in B' is adjacent to a vertex in P_1 then it is not adjacent to any vertex in P_2, and if a vertex in B' is adjacent to a vertex in P_2 then it is not adjacent to any vertex in P_1. We have two paths P_1 and P_2. In the following steps, we will continue to divide the paths into smaller and smaller paths.
Step 3. Let R_j denote the set of vertices of B in P_j that are adjacent to some vertices of A not in P_j. Let $R = \cup R_j$. If R is empty, then the procedure stops.
Step 4. Let $r = \max R$. If R is not empty, remove r from G. The path containing r is split into two paths. This path is therefore replaced by the two new paths. The number of paths is increased by 1 after the split. Go to Step 3.

Let C be the set of vertices in B removed by the above procedures. We can see that there are $|C|+1$ paths when the above procedure terminates. Note that the starting vertex and ending vertex of each path are the vertices in A. Let the set of paths are sorted in increasing order of their starting vertices. The following three properties hold throughout the procedure:
(1) Each vertex of B in P_j is not adjacent to any vertex in P_f for $j < f$.
(2) Each vertex of F_1 is not adjacent to any vertex in $P_j \setminus R_j$ for all paths P_j.
(3) Let $N(b)$ denote the set of vertices adjacent to a vertex $b \in B'$. Then, $N(b) \subseteq P_j$ for some path P_j.
When the above procedure terminates, $R = \emptyset$. By the above properties, clearly, no two vertices in different paths are adjacent and no vertex in B' is adjacent to two vertices in different paths. Each path P_i and vertices in B' that are adjacent to a vertex in P_i form a set in the remaining graph, denoted by P_i'. P_1', P_2', $\cdots$, $P_{|C|+1}'$, F_1 are $|C|+2$ disjoint sets in $G \setminus C$. By the pigeonhole principle, G cannot have a complete path. ∎

A *Hamiltonian path* (HP) in a graph G is a path that passes through each node exactly once. We state several useful properties.

Property 3.2 *G has no Hamiltonian path if G has no complete path.*

Property 3.3 *Let r be an arbitrary vertex in G. G has no Hamiltonian circuit if $G \setminus \{r\}$ has no Hamiltonian path.*

Let the path produced by Algorithm Greedy-Path-CBG be called *greedy complete path*. Let G' be a subgraph of G by removing the first (left-most) edge of each $b \in B$ in G.

Lemma 3.4 *G' has a matching of size $|A|-1$ iff G has a complete path on A.*

Proof. Note that a_1 is isolated in G'. If G' has a matching of size $|A|-1$, clearly, $a_1, match(a_2), a_2, \cdots, match(a_n), a_n$ is a path in G that connects all vertices in A using $|A|-1$ vertices from B. By Lemma 3.1, G has a complete path on A. Conversely, if G has a complete path, $a_1, b_{i_1}, a_2, \cdots, b_{i_{n-1}}, a_n$, b_{i_k} is matched to a_{k+1} in G'. Hence, G' has a matching of size $|A|-1$. ∎

Property 3.5 *Let $a_1, b_{i_1}, a_2, \cdots, b_{i_{n-1}}, a_n$ be a greedy complete path in G. b_{i_k} matches a_{k+1} for $1 \leq k < n$ in G' by Glover's algorithm.*

Theorem 3.6 *A greedy complete path on A in a CBG $G = (A, B, E)$ can be found in $O(\log^2 n)$ time using n processors on an EREW PRAM.*

Proof. Since G' is a subgraph of G where $BEG[b, G'] = BEG[b, G] - 1$ and $END[b, G'] = END[b, G]$. G' can be obtained in $O(1)$ time using n processors. By Lemma 2.1, Dekel and Sahni's parallel matching algorithm produces the same matching as Glover's greedy algorithm, a greedy complete path on A in G can be found in the same time and processor complexity by Lemma 3.4 and Property 3.5. ∎

4 Hamiltonian circuit in a CBG

The necessary condition for a CBG $G = (A, B, E)$ to have an HC is $|A| = |B|$ and G has a complete path on A. Let $P = a_1, b_{i_1}, a_2, \cdots, b_{i_{n-1}}, a_n$ be a greedy complete path and $b_{i_n} = B \setminus \{b_{i_1}, \cdots, b_{i_{n-1}}\}$. If b_{i_n} is not adjacent to a_n, clearly, $G \setminus \{b_{i_{n-1}}\}$ has no complete path on A. Hence, G has no HC. From now on we assume that b_{i_n} is adjacent to a_n.

The idea of the HC algorithm is finding two subpaths, each of which begins at a_n and ends at a_1, the concatenation of the two paths forms an HC in G. We show the nonexistence of an HC, if such two subpaths cannot be found. To help identifying these two paths, we construct a directed graph $H = (V^H, E^H)$, where $V^H = A$ and edges are constructed as follows:

1. for $a_u, a_v \in A \setminus \{a_1, a_n\}$, $(a_u, a_v) \in E^H$ if $u > v$, $(b_{i_u}, a_v) \in E$, $BEG[b_{i_v}] < a_v$;
2. $(a_n, a_v) \in E^H$ if $v < n$, $(b_{i_n}, a_v) \in E$ and $BEG[b_{i_v}] < a_v$;
3. $(a_u, a_1) \in E^H$ if $u > 1$ and $(b_{i_u}, a_1) \in E$;

Let $P^H = a_{h_t}, a_{h_{t-1}}, \cdots, a_{h_1}$ be a path from a_n to a_1 in H, where $a_{h_t} = a_n$ and $a_{h_1} = a_1$. Let $P(a_i, a_j)$ denote a subpath of P starting from a_i reversely to a_j for $i > j$. For example, $P(a_5, a_2) = a_5, b_{i_4}, a_4, b_{i_3}, a_3, b_{i_2}, a_2$. We construct two paths P_1 and P_2 based on P and P^H. Consider the following cases.

Case 1. $t = 2$. In this case, we let
$P_1 = P(a_{h_t}, a_{h_1})$.
$P_2 = a_{h_t}, b_{i_{h_t}}, a_{h_1}$.

Case 2. $t = 3$. In this case, we let
$P_1 = P(a_{h_t}, a_{h_2+1}), b_{i_{h_2}}, a_{h_1}$, and
$P_2 = a_{h_t}, b_{i_{h_t}}, P(a_{h_2}, a_{h_1})$.

Case 3. $t > 3$. If t is even, (see Figure 5.1) let
$P_1 = P(a_{h_t}, a_{h_{t-1}+1}), b_{i_{h_{t-1}}}, P(a_{h_{t-2}}, a_{h_{t-3}+1}), b_{i_{h_{t-3}}}, \cdots, P(a_{h_4}, a_{h_3+1}), b_{i_{h_3}}, P(a_{h_2}, a_{h_1})$, and
$P_2 = a_{h_t}, b_{i_{h_t}}, P(a_{h_{t-1}}, a_{h_{t-2}+1}), b_{i_{h_{t-2}}}, P(a_{h_{t-3}}, a_{h_{t-4}+1}), b_{i_{h_{t-4}}}, \cdots, P(a_{h_3}, a_{h_2+1}), b_{i_{h_2}}, a_{h_1}$.
If t is odd, let
$P_1 = P(a_{h_t}, a_{h_{t-1}+1}), b_{i_{h_{t-1}}}, P(a_{h_{t-2}}, a_{h_{t-3}+1}), b_{i_{h_{t-3}}}, \cdots, P(a_{h_3}, a_{h_2+1}), b_{i_{h_2}}, a_{h_1}$
$P_2 = a_{h_t}, b_{i_{h_t}}, P(a_{h_{t-1}}, a_{h_{t-2}+1}), b_{i_{h_{t-2}}}, P(a_{h_{t-3}}, a_{h_{t-4}+1}), b_{i_{h_{t-4}}}, \cdots, P(a_{h_4}, a_{h_3+1}), b_{i_{h_3}}, P(a_{h_2}, a_{h_1})$.

In the following, we describe the algorithm and give two examples followed by the proof for the algorithm.

Algorithm: HC-CBG (Finding an HC in a CBG)
Input: A CBG $G = (A, B, E)$ in compact form.
Ouput: An HC in G, or
a message stating that G has no HC.

1 Finding a greedy complete path on A in G.
If not found, report no HC, exit;
2 Construct H from P;
3 Find a path P^H in H from a_n to a_1;
If not found, report no HC, exit;
4 Obtain P_1 and P_2 and establish an HC.

Example 1: For the graph in Figure 5.2, $P = a_1, b_1, a_2, b_3, a_3, b_2, a_4, b_4, a_5, b_5, a_6, b_6, a_7, b_7, a_8$. We construct H shown in Figure 5.3 (Note we omitted isolated vertices in H). If we select $P^H = a_8, a_5, a_2, a_1$, an HC is found by joining $P_1 = P(a_8, a_6), b_5, P(a_2, a_1) = a_8, b_7, a_7, b_6, a_6, b_5, a_2, b_1, a_1$ with $P_2 = a_8, b_8, P(a_5, a_3), b_3, a_1 = a_8, b_8, a_5, b_4, a_4, b_2, a_3, b_3, a_1$. If we select $P^H = a_8, a_6, a_5, a_2, a_1$, an HC is found by joining $P_1 = P(a_8, a_7), b_6, P(a_5, a_3), b_3, a_1 = a_8, b_7, a_7, b_6, a_5, b_4, a_4, b_2, a_3, b_3, a_1$ with $P_2 = a_8, b_8, P(a_6, a_6), b_5, P(a_2, a_1) = a_8, b_8, a_6, b_5, a_2, b_1, a_1$.

Example 2: For the graph in Figure 5.4, $P = a_1, b_1, a_2, b_3, a_3, b_2, a_4, b_4, a_5, b_5, a_6, b_6, a_7, b_7, a_8$. Clearly, a_8 is not adjacent to any vertex in H, Thus, P^H cannot be found. Hence, G has no HC.

A vertex a_u is said to be *reachable in H* if there exists a path from a_n to a_u in H.

Property 4.1 *If $(a_u, a_w) \in E^H$, then $(a_u, a_v) \in E^H$ for every vertex a_v between a_w and a_u in P with $BEG[b_{i_v}] < a_v$.*

Property 4.2 *If a_u is reachable from a_n in H, then a_v is reachable from a_n in H for every a_v between a_u and a_n in P with $BEG[b_{i_v}] < a_v$.*

Lemma 4.3 *G has an HC iff H has a path from a_n to a_1.*

Proof. If H has a path P^H from a_n to a_1, by the algorithm, we can construct P_1 and P_2 to form an HC. Suppose that H does not have a path from a_n to a_1. Let a_u be a reachable vertex in H with the smallest $BEG[b_{i_u}]$ value and $a_w = BEG[b_{i_u}]$. If there exists a vertex b_{i_v} of B between a_w and a_u in P having an BEG value smaller than a_w (i.e. $BEG[b_{i_v}] < a_w$), then $(a_u, a_v) \in E^H$. Thus, a_v is reachable from a_n in H and a_v is a vertex in H with the smaller $BEG[b_{i_v}]$ than $BEG[b_{i_u}](= a_w)$, a contradiction. Hence, no vertex of B between a_w and a_u in P has an BEG value smaller than a_w. If there exists a vertex b_{i_v} of B between a_u and a_n in P having an BEG value smaller than a_w, then $(a_n, a_v) \in E^H$ according to Property 4.2. This also contradicts that a_u has the smallest $BEG[b_{i_u}]$

value. Therefore, no vertex of B between a_w and a_n in P has an BEG value smaller than a_w. Hence, $b_{i_1}, \cdots, b_{i_{w-1}}$ are the only possible vertices for linking $a_1, \cdots, a_{w-1}, a_w$. This implies that $G \setminus \{b_{i_{w-1}}\}$ does not have a complete path on A. Therefore, G has no HC. ∎

Note that P^H can be any path in H from a_n to a_1. Thus, it is sufficient to find one such path. To speed up the computation of finding a path P^H, we introduce H', a subgraph of H, defined as follows. $(a_u, a_v) \in E^H$ $(a_v < a_u)$ is in H' iff $BEG[b_{i_v}]$ is the smallest among all a_v adjacent to a_u in H.

Lemma 4.4 *H has a path from a_n to a_1 iff H' has a path from a_n to a_1. Furthermore, such a path P^H can be found in $O(\log n)$ time using n^2 processors on an EREW PRAM.*

Proof. If H' has a path from a_n to a_1, clearly, this path is also in H. Suppose that H has a path from a_n to a_1. Let a_u be the vertex adjacent to a_n with the smallest $BEG[b_{i_u}]$ value and a_w be any vertex adjacent to a_n. Then (i) for any vertex a_f with $a_f < a_w$ and $a_f < a_u$, if $(a_w, a_f) \in E^H$, then $(a_u, a_f) \in E^H$; (ii) for any vertex a_f with $a_f < a_u$, if a_f is reachable from a_w, it is also reachable from a_u; (iii) if there exists a path from a_n to a_1 through a_w, there also exists a path from a_n to a_1 through a_u, not including a_w. Hence, after eliminating edge (a_n, a_w) from H, a path from a_n to a_1 still exists. For the same reason, we can eliminate all $(a_u, a_v) \in E^H$ $(a_v < a_u)$ except the one with the smallest $BEG[b_{i_v}]$. The result of the eliminations is H'. Therefore, H' has a path from a_n to a_1.

To construct H', for each a_u, we examine every a_v with $(b_{i_u}, a_v) \in E$ and find an a_v with the smallest $BEG[b_{i_v}]$ value, this takes $O(\log n)$ time using n processors on an EREW PRAM. Hence, constructing entire H' takes $O(\log n)$ time using n^2 processors on an EREW PRAM. Since H' is a forest, a path from a_n to a_1 can be found in $O(\log n)$ time using n processors. Therefore, P^H can be found in $O(\log n)$ time using n^2 processors. ∎

Theorem 4.5 *Algorithm HC-CBG finds an HC in a CBG $G = (A, B, E)$ in $O(\log^2 n)$ time using n^2 processors on an EREW PRAM.*

Proof. The correctness is established in Lemma 4.3. Finding a greedy complete path takes $O(\log^2 n)$ time using n processors by Lemma 3.6. Finding P^H takes $O(\log n)$ time using n^2 processors by Lemma 4.4. Therefore, the algorithm can be implemented in $O(\log^2 n)$ time using n^2 processors. ∎

5 Hamiltonian circuits in a CCBG

The necessary condition for a CCBG $G = (A, B, E)$ to have an HC is $|A| = |B|$. Several definitions are useful to establish our algorithm. Two vertices of $A' \subseteq A$ are said to be *consecutive* in A' if one of the vertices is the next vertex clockwise from the other in A'. a_i is *clockwise connectable* to a_j by $b \in B$ if b is adjacent to every vertex from a_i to a_j clockwise. a_i is *clockwise connected* to a_j in a path by $b \in B$ if b is adjacent to every vertex from a_i to a_j clockwise. A path is called a *sequential path* of A', if (i) it connects all A' vertices using $|A'| - 1$ vertices from B; (ii) every $b \in B$ clockwise connects a pair of two consecutive vertices of A'. (i.e. the path connects all vertices of A' consecutively in clockwise order starting from some $a_i \in A'$, and there is only one pair not connected in the path.) A sequential path is *complete* if $A' = A$. A Hamiltonian circuit is called a *sequential HC* if every pair of two consecutive vertices of A is clockwise connected by a vertex in B. We show that if a CCBG G has an HC, then G has a sequential HC.

Lemma 5.1 *Let A' be a subset of A and P_1 be a path that connects vertices in A' using $|A'| - 1$ vertices from B. Then P_1 can be transformed into a sequential path of A'.*

Proof. Clearly, the lemma holds for $|A'| = 1$ and $|A'| = 2$. Assume it is true for $|A'| = i$. Let $|A'| = i+1$ and $P_1 = P_1', b_v, a_u$ where P_1' is a partial path of P_1 that connects i vertices in A' using $i - 1$ vertices in B. By the induction hypothesis, P_1' can be transformed into a path P_2' that clockwise connects adjacent vertices of P_1' in A' using the same vertices of P_1' in B. Let P_2' be $a_{t_1}, b_{j_{t_1}}, \cdots, b_{j_{t_{i-1}}}, a_{t_i}$. Note that $a_{t_1}, \cdots, a_{t_i}$ are in clockwise order from a_{t_1}. Let a_{t_r} denote the last vertex in P_1'. a_{t_r} is connected to b_v in P_1. Consider two cases.

Case 1. a_u lies clockwise between a_{t_i} and a_{t_1}. If a_{t_r} is clockwise connectable to a_u by b_v, then we obtain P_2 as P_2', b_v, a_u. If a_u is clockwise connectable to a_{t_r} via b_v, then we obtain P_2 as a_u, b_v, P_2'.

Case 2. a_u lies clockwise between a_{t_h} and $a_{t_{h+1}}$ for some $h \neq i$. If a_{t_r} is clockwise connectable to a_u by b_v, then we obtain P_2 as $a_{t_1}, b_{j_{t_1}}, \cdots, a_{t_h}, b_v, a_u, b_{j_{t_h}}, a_{t_{h+1}}, \ldots, b_{j_{t_{i-1}}}, a_{t_i}$. If a_u is clockwise connectable to a_{t_r} by b_v, then we obtain P_2 as $a_{t_1}, b_{j_{t_1}}, \cdots, a_{t_h}, b_{j_{t_h}}, a_u, b_v, a_{t_{h+1}}, \ldots, b_{j_{t_{i-1}}}, a_{t_i}$.

Therefore, P_1 can be transformed into P_2 that clockwise connects consecutive vertices in A' using the same $|A'| - 1$ vertices from B. ∎

Corollary 5.2 *If P_1 is a path connecting all vertices in A using $|A| - 1$ vertices from B, then P_1 can be transformed into a complete sequential path.*

Our algorithm makes use of the minimal path cover on A in a CCBG. A *path cover* on A is a set of vertex-disjoint paths in G such that (i) each path has more than one vertex from A than from B; (ii) it covers all vertices in A. A *minimal path cover on A* is a path cover of minimal cardinality.

Lemma 5.3 *An arbitrary minimal path cover T on A in a CCBG can be transformed into a new minimal path cover T' on A such that every vertex of B in T' connects two consecutive vertices of A.*

Proof. By Lemma 5.1, any path in T can be transformed into a sequential path. Hence, we assume each path in T is a sequential path. For convenience, we call vertices of B in T *connectors* to link two vertices of A. We perform the following procedure to rearrange connectors. (Note T always has the same number of paths throughout the procedure, but paths may change due to rearrangement of connectors.)

Step 1. If every path in T is a sequential path, then the procedure stops. Otherwise, T has a path $P_i = a_{i_1}, b_{j_1}, a_{i_2}, \cdots, b_{j_{k-1}}, a_{i_k}$ with at least one connector that is not connecting two consecutive vertices. Let b_{j_h} be the first such connector in P_i. Then a_{i_h+1} is not in P_i. Let a_{i_h+1} be in path P_j. Consider two cases:

1. a_{i_h+1} is the first vertex in P_j. Then simply use b_{j_h} to connect a_{i_h} with a_{i_h+1}.
2. a_{i_h+1} is not the first vertex in P_j. Let b' clockwise connects a' with a_{i_h+1} in P_j. Since all vertices of A in P_i before a_{i_h} are consecutively connected, a_{i_1}, a_{i_2}, $\cdots$, a_{i_h} are located clockwise between a' and a_{i_h+1}. Thus, b' is adjacent to $a_{i_1}, a_{i_2}, \cdots, a_{i_h}$. Use b' to connect a' with a_{i_1} and b_{j_h} to connect a_{i_h} with a_{i_h+1}.

Step 2. Go to Step 1.

We call a pair of two consecutive vertices of A linked by connector in a path *a consecutive connection.* Step 1 increases consecutive connections by at least 1 (possibly 2 if a' and a_{i_1} are consecutively connected in Case 2.) Once a connector is used for a consecutive connection, it will never be rearranged. Therefore, this procedure eventually terminates when all connectors are used for consecutive connections. ∎

We assume that no vertex in B is connected to all vertices in A. We will handle the case when G has vertices in B adjacent to all vertices in A later. Further assume that the graph $G = (A, B, E)$ is not a CBG. To test whether a CCBG is CBG, one merely searches for a pair of consecutive vertices in A that is not adjacent to any vertex in B. If such a pair exists, G is a CBG. Let G' be a subgraph of G by removing the first clockwise edge of each $b \in B$ in G.

Lemma 5.4 *G has an HC iff G' has a complete matching.*

Proof. (if part) If G' has a complete matching, then an HC in G is $a_1, matched(a_2), a_2, matched(a_3), \cdots, a_{n-1}, matched(a_n), a_n, matched(a_1)$, where $matched(a_i)$ denotes a vertex matched to a_i in G'.

(only if part) If G has an HC, denoted by H_1, starting from a vertex in A and ending at a vertex in B, let this HC be P, b, where P is an arbitrary path connecting all vertices in A using $B \setminus \{b\}$ and b connects the first vertex in P with the last one in P. By Corollary 5.2, P can be transformed into a complete sequential path P'. Without loss of generality, let $P' = a_1, b_{j_1}, a_2, b_{j_2}, \cdots, a_{n-1}, b_{j_{n-1}}, a_n$. If a_n and a_1 are adjacent to b, clearly, G' has a complete matching, where a_i matches $b_{j_{i-1}}$ and a_1 matches b.

Assume that a_n and a_1 are not adjacent to b. Let R denote the set of vertices that are adjacent to a_n and a_1. Such R is nonempty since we assume that G is not a CBG. Let T denote a minimal path cover in $G \setminus R$. By Lemma 5.3, we assume that T is a minimal path cover on A such that every vertex of B in T connects two consecutive vertices of A. Since G has an HC, $|T| = |R|$. Then every vertex in $B \setminus R$ is used to connect two vertices of A in T. We call $b_{j_{i-1}}$ that connects a_{i-1} and a_i in P' a *designated connector* for a_{i-1} and a_i. We perform the following procedure to rearrange the vertices of $B \setminus R$ in T.

Step 1: If there exists a pair of unconnected consecutive vertices, (a_{i-1}, a_i), in T, such that the designated connector of (a_{i-1}, a_i), $b_{j_{i-1}}$, is not in R, then $b_{j_{i-1}}$ currently connects (a_{k-1}, a_k) for some k in T, disconnect (a_{k-1}, a_k) and use $b_{j_{i-1}}$ to connect (a_{i-1}, a_i); otherwise, exit. go to Step 1.

A pair of consecutive vertices in T once is connected by a designated connector, this connector will never be rearranged in the procedure. We call such a pair that is connected by a designated connector a *designated pair.* Since Step 1 increases the number of designated pairs by 1, the above procedure eventually will terminate. When it terminates, the designated connector of every pair of unconnected consecutive vertices in T is in R.

Since the number of pairs of unconnected consecutive vertices is $|R| - 1$ in T, they can be connected by $|R| - 1$ vertices in R. The remaining vertex in R can be used to connect a_n with a_1. Therefore, H_1 can be transformed into a sequential HC. ∎

If G has an HC, then $G \setminus \{b\}$ has a complete sequential path for any $b \in B$. Assume there is a vertex $b \in B$ adjacent to all vertices in A. Suppose (a_{i-1}, a_i) is the only pair not connected in a sequential path in $G \setminus \{b\}$. Then b can be used to clockwise connect a_{i-1} with a_i. In general, we can obtain the following lemma.

Lemma 5.5 *Let U be the set of vertices adjacent to all vertices in A with $|U| > 0$ and G' be a subgraph of $G \setminus U$ by removing the first clockwise edge of each $b \in B$ in $G \setminus U$. Then G has an HC iff $|M| = |A| - |U|$, where M is a MM in G'.*

Proof. If $|M| = |A| - |U|$, an HC in G can be obtained as follows: (i) if (b, a_i) is matched in M, then (a_{i-1}, b) and (b, a_i) are edges in the path; (ii) any unconnected pair of consecutive vertices in A can be connected using a vertex in U.

Assume that G has an HC. Then $G \setminus U$ has a minimal path cover of cardinality $|U|$. By Lemma 5.3, there exists a minimal path cover $T = \{P_1, \cdots, P_{|U|}\}$, in which every vertex of B connects two consecutive vertices of A. Let P_i be $a_{i_1}, b_{j_1}, \cdots, b_{j_{k-1}}, a_{i_k}$. Then $b_{j_{k-1}}$ can be matched to a_{i_k} in G'. Thus, every vertex in P_i is matched except a_{i_1}. Therefore, every vertex in A is matched except the

first vertex in each path of T. Clearly, $|M| = |A| - |U|$. ∎

We give the detailed algorithm as follows.

Algorithm: HC-CCBG (Finding an HC in a CCBG)
Input: A CCBG $G = (A, B, E)$ with $|A| = |B|$ and G is not a CBG.
Ouput: An HC in G, or a message stating that G has no HC.

1 Let U be the set of vertices adjacent to all vertices in A;
2 Let G' be a subgraph of $G \setminus U$ by removing the first clockwise edge of each $b \in B$ in $G \setminus U$;
3 **if** $U = \emptyset$ **then**
4 **if** G' has a MM of size $|A|$ **then**
5 An HC is $a_1, b_{i_1}, a_2, b_{i_2}, \cdots, a_{n-1}, b_{i_{n-1}}, a_n, b_{i_n}$,
6 where $b_{i_{k-1}}$ matches a_k for $1 < k \leq n$ and b_{i_n} matches a_1;
7 **else** G has no HC;
8 **end if**;
9 **else** {U is not empty}
10 Find M, a MM in G',
11 **if** $|M| + |U| = |A|$ **then**
12 G has an HC constructed as follows:
11 if (b, a_i) is matched in M, then (a_{i-1}, b) and (b, a_i) are edges in the path;
12 any unconnected pair of consecutive vertices can be connected using a vertex in U;
13 **else**
14 G has no HC;
15 **end if**;
16 **end if**

Theorem 5.6 *An HC in a CCBG $G = (A, B, E)$ can be found in $O(\log^2 n)$ time using n processors on an EREW PRAM.*

Proof. If $|U| = \emptyset$, the correctness for this case is established by Lemma 5.4. Since G' can be identified in $O(1)$ time using n processors and a MM in G' can be found in $O(\log^2 n)$ time using n processors by Lemma 2.2, an HC can be found in $O(\log^2 n)$ time using n processors.

If $|U| \neq \emptyset$, the correctness for this case is established by Lemma 5.5. Similarly, the complexity for finding an HC in this case is also $O(\log^2 n)$ time and n processors. ∎

References

[1] A. A. Bertossi and S. Moretti. Parallel algorithms on circular-arc graphs. *Information Processing Letters* **33(6)** (1990) 275–281.

[2] M-S Chang, Y-C Tsai and Y. D. Liang. Efficient Algorithms for the Domination and Steiner Tree Problems on Convex Bipartite Graphs. Manuscript, 1995.

[3] E. Dekel and S. Sahni. A parallel matching algorithm for convex bipartite graphs and applications to scheduling. *J. of Parallel and Distributed Computing* **1** (1984) 185–205.

[4] E. Dekel and S. Sahni. Binary tree and parallel scheduling algorithms. *IEEE Transactions on Computers* **33** (3) (1983) 307–315.

[5] Faith E. Fich. The complexity of computation on the parallel random access machine. In Reif [16], chapter 20, pages 843–899.

[6] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman and Company, San Fransisco, 1979.

[7] F. Glover. Maximum matching in convex bipartite graphs. *Naval Res. Logist. Quart.* **14** (1967) 313-316.

[8] R. Greenlaw, H. J. Hoover, and W. L. Ruzzo. *Topics in Parallel Computation: A Guide to the Theory of $\mathcal{P}$-completeness.* Oxford University Press, New York, to appear.

[9] C-W. Ho and R. C. T. Lee. Efficient parallel algorithms for finding maximum cliques, clique trees, and minimum coloring on chordal graphs. *Information Processing Letters* **28(6)** (1988) 301–309.

[10] R. M. Karp and V. Ramachandran. Parallel algorithms for shared-memory machines. In van Leeuwan [11], chapter 17, pages 869–941.

[11] Jan van Leeuwan, editor. *Handbook of Theoretical Computer Science*, volume A: Algorithms and Complexity. M.I.T. Press/Elsevier, 1990.

[12] Y. D. Liang and N. Blum. Circular convex bipartite graphs: maximum matching and Hamiltonian circuit. *Information Processing Letters*, **56** (1995) 215-219.

[13] Y. D. Liang, G. K. Manacher, C. Rhee, and T. A. Mankus. An $O(n)$ algorithm for finding Hamiltonian paths and circuits in circular-arc graphs. Manuscript, August, 1992.

[14] R. M. Karp. Reducibility among combinatorial problems, in R. E. Miller and J. W. Thacther, eds., *Complexity of Computer Computations*, Plenum Press, New York, (1972) 85–103.

[15] J. Naor, M. Naor, and A. A. Schäffer. Fast parallel algorithms for chordal graphs. *Proceedings 19th Ann. ACM Symposium on Theory of Computing* (1987) 355–364.

[16] John H. Reif, editor. *Synthesis of Parallel Algorithms.* Morgan Kaufman, San Mateo, CA, 1993.

[17] W. Shih, T. C. Chern, and W. L. Hsu. An $O(n^2 \log n)$ algorithm for Hamiltonian cycle problem on circular-arc graphs. *SIAM J. Computing* **21** (6) (1992) 1026-1046.

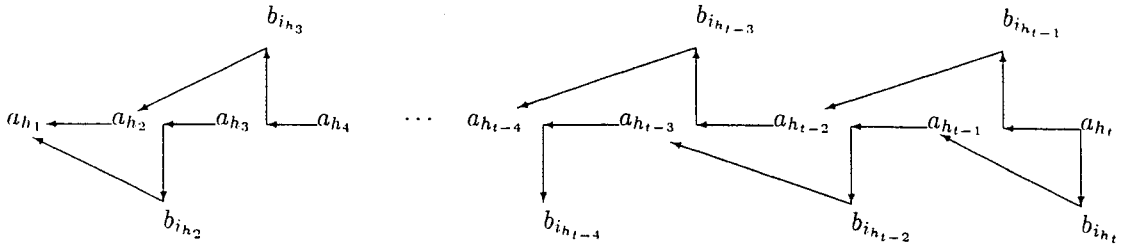

Figure 5.1: Identifying two paths P_1 and P_2.

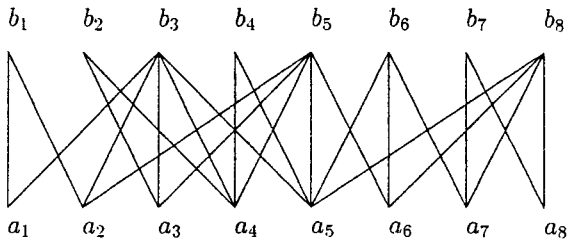

Figure 5.2: A CBG G having an HC.

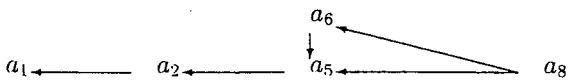

Figure 5.3: Illustration of H constructed for Figure 5.2.

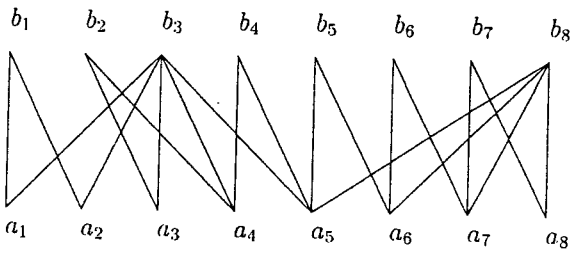

Figure 5.4: A CBG having no HC.