The Hamiltonian Cycle Problem on Circular-Arc Graphs*

Ruo-Wei Hung†‡, Maw-Shang Chang§, and Chi-Hyi Laio

Abstract—**A Hamiltonian cycle in a graph G is a simple cycle in which each vertex of G appears exactly once. The Hamiltonian cycle problem involves testing whether a Hamiltonian cycle exists in a graph, and finds one if such a cycle does exist. It is well known that the Hamiltonian cycle problem is one of the classic NP-complete problems on general graphs. Shih et al. solved the Hamiltonian cycle problem on circular-arc graphs in $O(n^2 \log n)$ time [36], where n is the number of vertices of the input graph. Whether there exists a more efficient algorithm for solving the Hamiltonian cycle problem on circular-arc graphs has been opened for a decade. In this paper, we present an $O(\Delta n)$-time algorithm to solve it, where Δ denotes the maximum degree of the input graph.**

Keywords: graph algorithms, Hamiltonian cycle problem, path cover problem, interval graphs, circular-arc graphs

1 Introduction

All graphs considered in this paper are finite and undirected, without loops or multiple edges. Let $G = (V, E)$ be a graph with vertex set V and edge set E. Throughout this paper, let m, n, and Δ denote the number of edges, the number of vertices, and the maximum degree of G, respectively. A *Hamiltonian cycle* in G is a simple cycle in which each vertex of G appears exactly once. A *Hamiltonian path* in G is a simple path with the same property. The *Hamiltonian cycle* (*resp. path*) *problem* involves testing whether or not G contains a Hamiltonian cycle (resp. path), and finds one if such a cycle (resp. path) does exist. A graph G is said to be *Hamiltonian* if it contains a Hamiltonian cycle. The *Hamiltonian problems* include the Hamiltonian path and Hamiltonian cycle problems and have numerous applications in different areas, including establishing transport routes, production launching, the on-line optimization of flexible manufacturing systems [2], computing the perceptual boundaries of dot patterns [33], and pattern recognition [3, 34, 37]. It is well known that the Hamiltonian problems are NP-complete for general graphs [15, 25]. The same holds true for bipartite graphs [26], split graphs [16], circle graphs [9], undirected path graphs [4] and grid graphs [24]. However, polynomial time algorithms exist for the Hamiltonian cycle or Hamiltonian path problem on some special classes of graphs, such as interval graphs [1, 7], permutation graphs [13, 35], cocomparability graphs [10, 12], and distance-hereditary graphs [17, 20, 22]. A *path cover* of a graph G is a family of vertex-disjoint paths that covers all vertices of G. Given a graph G, the *path cover problem* is to find a path cover of G of minimum number, denoted by $\pi(G)$, of paths. This problem is NP-hard for general graphs [15] since it contains the Hamiltonian path problem as a special case.

A graph $G = (V, E)$ is called an *intersection graph* for a finite family $\mathcal{F}$ of nonempty sets if there is a one-to-one correspondence between $\mathcal{F}$ and V such that two sets in $\mathcal{F}$ have nonempty intersection if and only if their corresponding vertices in V are adjacent. We call $\mathcal{F}$ an *intersection model* of G. For an intersection model $\mathcal{F}$, we use $G(\mathcal{F})$ to denote the intersection graph for $\mathcal{F}$. If $\mathcal{F}$ is a family of intervals on a real line, then G is called an *interval graph* for $\mathcal{F}$ and $\mathcal{F}$ is called an *interval model* of G. If $\mathcal{F}$ is a family of arcs on a circle, then G is called a *circular-arc graph* for $\mathcal{F}$ and $\mathcal{F}$ is called a *circular-arc model* of G. The two classes of interval graphs and circular-arc graphs have a variety of applications involving traffic light sequencing, VLSI design, scheduling [16], and genetics [39].

Shih et al. solved the Hamiltonian cycle problem on circular-arc graphs in $O(n^2 \log n)$ time [36]. Whether there exists an efficient algorithm whose time-complexity is better than $O(n^2 \log n)$ for solving the Hamiltonian cycle problem on circular-arc graphs has been opened for a decade. In this paper, we present an $O(\Delta n)$-time algorithm to solve it. We explain our strategy by briefly describing the algorithm as follows. It reduces the problem to the Hamiltonian cycle problem and the path cover problem on interval graphs and uses an $O(n)$-time algorithm proposed by Chang et al. [7] for interval graphs. Note that interval graphs form a proper subclass of circular-arc graphs. The basic idea is described in the following. We first convert the circular-arc model F to an interval model I, where $G(I)$ is a spanning subgraph of $G(F)$. If $G(I)$ is Hamiltonian, then $G(F)$ is Hamiltonian. If $G(I)$ is not Hamiltonian, then a subset C of F is found such that $G(F \setminus C)$ has either $|C|$ or $|C| + 1$

*The research was supported in part by the National Science Council of Taiwan under grant no. NSC95-2221-E-324-056.

†Department of Computer Science and Information Engineering, Chaoyang University of Technology, Wufong, Taichung 413, Taiwan.

‡Corresponding author's e-mail: rwhung@cyut.edu.tw

§Department of Computer Science and Information Engineering, National Chung Cheng University, Ming-Hsiung, Chiayi 621, Taiwan.

connected components. If $G(F \setminus C)$ has $|C|+1$ connected components, then $G(F)$ is not Hamiltonian by pigeonhole principle. Otherwise, a nonempty subset E^C of edge set of $G(F)$ is found. Every edge e of E^C connects two arcs of C and $G(F')$ is obtained by removing E^C from $G(F)$. Furthermore, $G(F')$ is still a circular-arc graph. By pigeonhole principle, every edge e of E^C will not be in any Hamiltonian cycle of $G(F)$. Thus, $G(F)$ is Hamiltonian if and only if $G(F')$ is Hamiltonian. Let $F = F'$ and repeat the above process until the problem is solved.

Related works are summarized below. Arikati and Rangan presented an $O(n + m)$-time algorithm to solve the path cover problem on interval graphs [1]. Chang et al. proposed $O(n)$-time algorithms for both the Hamiltonian cycle and path cover problems on interval graphs given an interval model that is a set of n sorted intervals [7]. Damaschke presented an $O(n^5)$-time algorithm to solve the Hamiltonian path problem on circular-arc graphs [11]. Shih et al. proposed an $O(n^2 \log n)$-time algorithm for the Hamiltonian cycle problem on circular-arc graphs [36]. The algorithm proposed by Bonuccelli and Bovet for solving the path cover problem on circular-arc graphs [5] contains a flaw which was pointed out in [36]. Some researchers [6, 28, 29] claimed that $O(n)$-time algorithms exist for the Hamiltonian cycle problem and the path cover problem on circular-arc graphs given a circular-arc model that is a set of n sorted arcs, but they have not yet succeeded in proving the correctness of their algorithms. In [23], we presented an $O(n)$-time approximation algorithm for the path cover problem on circular-arc graphs. We showed that the cardinality of the path cover found by the approximation algorithm is at most one more than the optimal one. By using the result, we reduced the path cover problem on circular-arc graphs to the Hamiltonian cycle problem on the same class of graphs in $O(n)$ time.

The paper is organized as follows. In Section 2, we establish the notation and related terminology, and review a greedy algorithm for the path cover problem on interval graphs. In Section 3, we present an $O(\Delta n)$-time algorithm to solve the Hamiltonian cycle problem on circular-arc graphs. Section 4 reveals how to prove the correctness of our algorithm, analyzes the complexity of our algorithm, and introduces a technique that reduces the related problems to the Hamiltonian cycle problem on circular-arc graphs. Finally, in Section 5 we conclude the paper and discuss possible future researches.

2 Preliminaries

In this section, we establish basic terminology and review a greedy algorithm for solving the path cover problem on interval graphs.

Let $G = (V, E)$ be a graph without isolated vertices. A path P in G, denoted by $v_1 \rightarrow v_2 \rightarrow \cdots \rightarrow v_{|P|-1} \rightarrow v_{|P|}$, is a sequence $(v_1, v_2, \cdots, v_{|P|-1}, v_{|P|})$ of vertices so that two vertices are adjacent if and only if they are consecutive in the sequence. The first and last vertices visited by P are called the *path-start* and *path-end* of P, denoted by $start(P)$ and $end(P)$, respectively. We call v_i the predecessor of v_{i+1}, denoted by $v_i = predecessor(v_{i+1})$, in P, for $1 \leq i \leq |P| - 1$. We use $v_i \in P$ to denote "P visits v_i". In addition, we will use P to refer to the set of vertices visited by path P if it is understood without ambiguity. For any $S \subseteq V$, $G \setminus S$ denotes the subgraph of G induced by $V \setminus S$, i.e., $G \setminus S = G[V \setminus S]$. A subset C of vertices of G is called a *cutset* if the removal of C from G disconnects G. We call C a *connecting set* of G if C is a cutset of G and the removal of C from G disconnects G into at least $|C| + 1$ connected components.

For $C \subseteq V$, let $E(C)$ denote the set of edges of E that connect two vertices in C. The following two propositions can be easily verified by the pigeonhole principle.

Proposition 2.1. *Let C be a cutset of a connected graph G and let g be the number of connected components in $G \setminus C$. If $g > |C|$, then G has no Hamiltonian cycle.*

Proposition 2.2. *Let C be a cutset of a connected graph G and let g be the number of connected components in $G \setminus C$ such that $g = |C|$. Then, G is Hamiltonian if and only if $G^C = (V, E \setminus E^C)$ is Hamiltonian, where E^C is any nonempty subset of $E(C)$.*

Our algorithm for the Hamiltonian cycle problem on circular-arc graphs reduces the problem to the path cover and Hamiltonian cycle problems on interval graphs. Arikati and Rangan presented an $O(n + m)$-time algorithm for the path cover problem on interval graphs [1]. Manacher et al. solved the Hamiltonian cycle problem on interval graphs in linear time [30]. These two algorithms can be implemented in $O(n)$ time given an interval model that is a set of n sorted intervals [7].

Now, we review the algorithm in [1] for the path cover problem on interval graphs. The algorithm is assumed that the input graph is given by an interval model I that is a set of n sorted intervals labeled by $1, 2, \cdots, n$ in increasing order of their right endpoints. Notice that we do not distinguish an interval from its label. The left endpoint of interval x is denoted by $left(x)$ and the right endpoint by $right(x)$. Interval x is denoted by ($left(x)$, $right(x)$). An interval x is said to *contain* another interval y if every point of y falls within the interior of ($left(x)$, $right(x)$). For convenience, we need the following notation: (1) For two distinct intervals x, y in I, x is smaller than y, denoted by $x < y$, if $right(y)$ is to the right of $right(x)$, and (2) $s(I)$ denotes the interval in I with the leftmost right endpoint, i.e., $s(I) \leq x$ for $x \in I$.

In the following, we review two algorithms presented in [1] and [23] on interval graphs. The first one is a greedy algorithm for the path cover problem on interval graphs. The second one is also a greedy algorithm for finding a

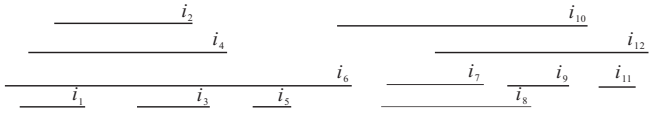

Fig. 1: An interval model I of twelve sorted intervals.

subset C of a set I of sorted intervals so that $G(I \setminus C)$ has at least $|C| + 1$ connected components.

The algorithm presented in [1] for the path cover problem on an interval model I uses a greedy principle to extend a path $Z_i = z_1 \to z_2 \to \cdots \to z_k$, $k \geq 1$. Initially, $i = 1$ and Z_1 visits $s(I)$, i.e., $start(Z_1) = s(I)$. Then, the unvisited neighbor of z_k with the leftmost right endpoint is chosen and Z_i is extended to visit it. If such a neighbor does not exist, then Z_i is stopped and a new path Z_{i+1} is started in the remaining graph from the smallest labeled unvisited interval if it is possible. To simplify the notation, we denote the algorithm and its output by Algorithm GP and $GP(I)$, respectively. For instance, given a set of sorted intervals shown in Fig. 1, Algorithm GP outputs two paths $Z_1 = i_1 \to i_2 \to i_3 \to i_4 \to i_6 \to i_5$ and $Z_2 = i_7 \to i_8 \to i_9 \to i_{10} \to i_{12} \to i_{11}$. The following theorem was given by Arikati et al. in [1] and shows the optimality of Algorithm GP.

Theorem 2.3. (*Arikati et al. [1]; Hung et al. [23]*) *$GP(I)$ is a minimum path cover of $G(I)$.*

It is easy to see that Algorithm GP runs in $O(n + m)$ time. Chang et al. showed that Algorithm GP can be done in $O(n)$ time and $O(n)$ space given a set of sorted intervals [7]. The paths in $GP(I)$ are called *greedy paths*. Note that the path-start of a greedy path is the smallest labeled interval in the path. Define $L(Z)$ of a path Z to be the collection of intervals in Z which are larger than $end(Z)$, i.e., $L(Z) = \{x | x \in Z \text{ and } end(Z) < x\}$. A path Z is called a *monotone path* if and only if $L(Z) = \emptyset$. For instance, let $Z_1 = i_1 \to i_2 \to i_3 \to i_4 \to i_6 \to i_5$ be a path of $G(I)$ shown in Fig. 1, then $L(Z_1) = \{i_6\}$. Given a greedy path Z, we can find a subset $C(Z)$ of Z so that the removal of $C(Z)$ from Z disconnects Z into $|C(Z)|+1$ subpaths. The procedure was given in [23]. For readers' convenience, we describe it as follows.

Procedure GCS
Input: Z, a greedy path in $GP(I)$.
Output: $C(Z)$, the *greedy connecting set* of path Z, and $R(Z)$, the set of all of the sub-paths of Z obtained by removing $C(Z)$ from Z.
Method:

1. $S = Z$; $C(Z) = \emptyset$; $R(Z) = \emptyset$;
2. $z = end(Z)$; $CurrentPathEnd = end(Z)$;
3. **while** $z \neq start(S)$ **do**
4. let $z = predecessor(z)$;
5. **if** $CurrentPathEnd < z$, **then**
6. let $S = S_1 \to z \to S_2$, where S_1 and S_2 are two subpaths of S;
7. $C(Z) = C(Z) \cup \{z\}$; $R(Z) = R(Z) \cup \{S_2\}$;
8. $S = S_1$; $z = CurrentPathEnd = end(S_1)$;
9. $R(Z) = R(Z) \cup \{S\}$;
10. **output** $C(Z)$ and $R(Z)$.

For instance, given a greedy path $Z_1 = i_1 \to i_2 \to i_3 \to i_4 \to i_6 \to i_5$ of Fig. 1, Procedure GCS outputs $C(Z_1) = \{i_6\}$ and $R(Z_1) = \{i_1 \to i_2 \to i_3 \to i_4, i_5\}$. Since each interval is visited once, Procedure GCS(Z) runs in $O(|Z|)$ time.

3 The Hamiltonian Cycle Algorithm

Circular-arc graphs are simple generalization of interval graphs. However, circular-arc graphs have rich structure, and there is a long history for the recognition problem. Tucker presented an $O(n^3)$-time algorithm for testing whether a graph is a circular-arc graph [38]. Hsu proposed an $O(mn)$-time algorithm to recognize circular-arc graphs [19]. Eschen and Spinrad proposed an $O(n^2)$-time recognition algorithm for circular-arc graphs [14]. Now, an $O(m + n)$-linear-time recognition algorithm for circular-arc graphs has been proposed by McConnell in [32]. A circular-arc model F can be obtained by these recognition algorithms in the affirmative case. Hence, researchers studying circular-arc graphs sometimes assumed that a set of arcs with endpoints sorted is given [7, 8, 18, 21, 31]. In the following, we assume that the input graph is given by a circular-arc model F that is a set of n sorted arcs.

An arc x in F that begins with endpoint p and ends at endpoint q in clockwise direction is denoted by (p, q). We call p the *head*, denoted by $h(x)$, and q the *tail*, denoted by $t(x)$, of arc (p, q). The contiguous part of the circle that begins with an endpoint c and ends at an endpoint d in clockwise direction is referred to as segment (c, d), denoted by $seg(c, d)$, of the circle. We use "*arc*" to refer to a member of F and "*segment*" to refer to a part of the circle between two endpoints. A point y on the circle is said to be contained in arc (or segment) (p, q) if it falls within the interior of $seg(p, q)$. An arc or segment x is said to *contain* another arc or segment y if x contains every point of y. Two arcs, or two segments, or an arc and a segment *intersect* if and only if they share a point. For a point q on the circle, let $B_{\rm p}(q)$ be the set of all arcs in F containing point q. For an arc x in F, let $B_{\rm a}(x)$ be the set of all arcs in F containing arc x. Without loss of generality, we assume that (1) all endpoints are distinct, (2) no arc covers the entire circle, and (3) an arc (segment) does not include its two endpoints, i.e., it is an open segment of the circle.

We first propose a procedure to map a set F of arcs into a set of intervals. Intuitively, this procedure cuts the cir-

cle from the head (resp. tail) of an arc v and stretches the circle into a horizontal line. Thus an arc becomes an interval. We stretch the circle in this way such that the head (resp. tail) of arc v is on the left hand side of the horizontal line. In other words, $h(v)$ (resp. $t(v)$) becomes the left endpoint of the corresponding interval of arc v. An arc that contains the cut point is not cut into two intervals but is stretched into an interval. Let $\mathcal{B}$ be a subset of $B_{\rm a}(v)$. Note that $\mathcal{B}$ is a part of the input of this procedure. Those arcs in $\mathcal{B}$ are stretched into intervals that contain the left endpoint of the corresponding interval of arc v. Those arcs that contain the cut point but are not in $\mathcal{B}$ are stretched into intervals that do not contain the left endpoint of the corresponding interval of arc v. The mapping procedure that cuts the circle from $h(v)$ (resp. $t(v)$) is called *clockwise* (resp. *counterclockwise*) mapping with respect to arc v and $\mathcal{B}$ and is formally presented as follows.

Procedure ArcToI
Input: A set F of n sorted arcs, an arc $v \in F$, and a subset $\mathcal{B}$ of $B_{\rm a}(v)$.
Output: A set $I_c(\mathcal{B}, v)$ (resp. $I_{cc}(\mathcal{B}, v)$) of n sorted intervals.
Method:

1. starting from point $h(v)$ (resp. $t(v)$), label the endpoints of arcs of F in clockwise (resp. counterclockwise) direction; that is, the endpoint located on point $h(v)$ (resp. $t(v)$) is labeled by 1;
2. let $\ell(p)$ denote the label of endpoint p;
3. **for** each arc $x \notin \mathcal{B}$, x is mapped to interval $I_c(x) = (h(x), t(x))$ if $\ell(t(x)) > \ell(h(x))$ (resp. $I_{cc}(x) = (t(x), h(x))$ if $\ell(h(x)) > \ell(t(x))$) and to interval $I_c(x) = (h(x), t(x) + 2n)$ (resp. $I_{cc}(x) = (t(x), h(x) + 2n)$) otherwise;
4. **for** each arc $x \in \mathcal{B}$, x is mapped to interval $I_c(x) = (1, t(x))$ (resp. $I_{cc}(x) = (1, h(x))$);
5. let $I_c(\mathcal{B}, v) = \{I_c(x) | x \in F\}$ (resp. $I_{cc}(\mathcal{B}, v) = \{I_{cc}(x) | x \in F\}$) and **output** $I_c(\mathcal{B}, v)$ (resp. $I_{cc}(\mathcal{B}, v)$).

Apparently, both $G(I_c(\mathcal{B}, v))$ and $G(I_{cc}(\mathcal{B}, v))$ are spanning subgraphs of $G(F)$. For instance, given a set F of arcs shown in Fig. 2, arc a_1, and $\mathcal{B} = B_{\rm a}(a_1)$, Procedure ArcToI first labels the endpoints of arcs of F (as shown in Fig. 3) in clockwise direction starting from $h(a_1)$. Then, it converts F into a set $I_c(B_{\rm a}(a_1), a_1))$ of intervals as shown in Fig. 4. On the other hand, Procedure ArcToI constructs a set $I_{cc}(\emptyset, a_6)$ (as shown in Fig. 5) of intervals for the set F of arcs shown in Fig. 2. Careful implementation of Procedure ArcToI takes $O(n)$ time and $O(n)$ space.

Let v be an arc in a set F of arcs, $\mathcal{B}$ be a subset of $B_{\rm a}(v)$, I_c and I_{cc} represent $I_c(\mathcal{B}, v)$ and $I_{cc}(\mathcal{B}, v)$, respectively, and let I be either I_c or I_{cc}. Let x be an arc in F and let

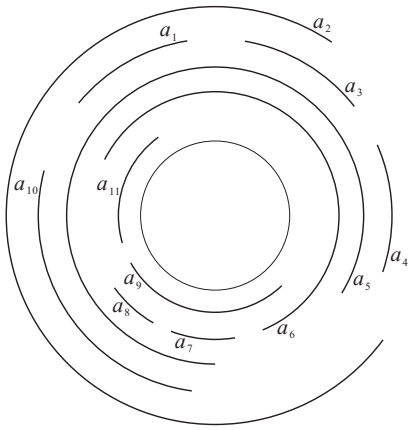

Fig. 2: A circular-arc model F of eleven sorted arcs.

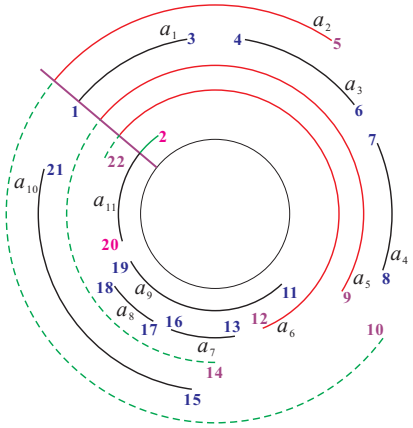

Fig. 3: The clockwise labeling by Procedure ArcToI given F shown in Fig. 2, arc a_1, and $\mathcal{B} = B_{\rm a}(a_1)$, where dashed lines indicate the removed segments of arcs from F.

X be a subset of F. We use $I(x)$ to denote the interval corresponding to arc x in I and $I(X)$ to denote $\{I(x) | x \in X\}$. Conversely, let y be an interval in I and let Y be a subset of I. We use $F(y)$ to denote the arc corresponding to y in F and $F(Y)$ to denote $\{F(y) | y \in Y\}$.

Now, we present an $O(\Delta n)$-time algorithm to solve the Hamiltonian cycle problem on circular-arc graphs. Our algorithm reduces the problem to the Hamiltonian cycle and the path cover problems on interval graphs. Note that the Hamiltonian cycle and the path cover problems on interval graphs can be solved in $O(n)$ time if the input is a set of n intervals with endpoints sorted [1, 7, 30]. The algorithm is formally presented as follows.

Algorithm HC(F)
Input: F, a set of sorted arcs.
Output: A Hamiltonian cycle of $G(F)$ if it is Hamiltonian.
Method:

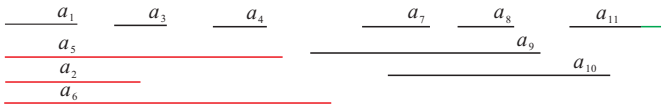

Fig. 4: The set $I_c(B_a(a_1), a_1)$ of intervals for the set F of arcs shown in Fig. 2.

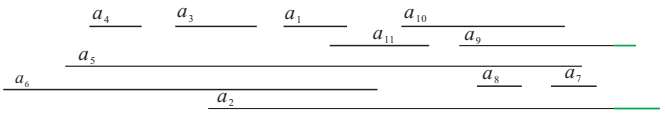

Fig. 5: The set $I_{cc}(\emptyset, a_6)$ of intervals for the set F of arcs shown in Fig. 2.

Step 1: Map F into a set I_c of intervals in a clockwise direction and test whether $G(I_c)$ is Hamiltonian.
(1.1) pick an arc μ of F that does not contain any other arc;
(1.2) map F into a set $I_c = I_c(B_a(\mu), \mu)$ of intervals by calling Procedure ArcToI that is a clockwise mapping w.r.t. $B_a(\mu)$ and μ;
(1.3) call Algorithm GP to compute $GP(I_c)$;
(1.4) let P be the first greedy path output by Algorithm GP(I_c) and let υ be the arc corresponding to $end(P)$;
(1.5) **if** $|GP(I_c)| = 1$ and arc μ intersects arc υ in $G(F)$, **then output** "$\upsilon \rightarrow P$" as a Hamiltonian cycle of $G(F)$ and **terminate** the algorithm.
(1.6) **if** $G(I_c)$ is Hamiltonian, **then output** the Hamiltonian cycle found in $G(I_c)$ and **terminate** the algorithm. /* call the algorithm in [7, 30] to test whether $G(I_c)$ is Hamiltonian or not */

Step 2: Map F into a set I_{cc} of intervals in a counterclockwise direction and compute $GP(I_{cc})$.
(2.1) call Procedure GCS(P) to find $L(P)$ and $C(P)$, where $P = P_1 \rightarrow c_1 \rightarrow P_2 \rightarrow \cdots \rightarrow c_{k-1} \rightarrow P_k \rightarrow c_k \rightarrow P_{k+1}$ and $C(P) = \{c_1, c_2, \cdots, c_k\}$;
(2.2) let υ_1 be the arc corresponding to $end(P_1)$;
(2.3) map F into a set $I_{cc} = I_{cc}(\emptyset, \upsilon_1)$ of intervals by calling Procedure ArcToI that is a counterclockwise mapping w.r.t. $\emptyset$ and υ_1;
(2.4) call Algorithm GP to compute $GP(I_{cc})$;

Step 3: Test whether $G(I_{cc})$ is Hamiltonian.
(3.1) **if** $|GP(I_{cc})| > 1$, **then output** "$G(F)$ has no Hamiltonian cycle" and **terminate** the algorithm.
(3.2) let Q be the only greedy path in $GP(I_{cc})$;
(3.3) let s and ω be the arcs corresponding to $start(Q)$ and $end(Q)$, respectively;
(3.4) **if** arc s intersects arc ω in $G(F)$, **then output** "$\omega \rightarrow Q$" as a Hamiltonian cycle of $G(F)$ and **terminate** the algorithm.
(3.5) **if** $G(I_{cc})$ is Hamiltonian, **then output** the Hamiltonian cycle found in $G(I_{cc})$ and **terminate** the algorithm. /* call the algorithm in [7, 30] to test whether $G(I_{cc})$ is Hamiltonian or not */
(3.6) call Procedure GCS(Q) to find $L(Q)$ and $C(Q)$, where $Q = Q_1 \rightarrow d_1 \rightarrow Q_2 \rightarrow \cdots \rightarrow d_{h-1} \rightarrow Q_h \rightarrow d_h \rightarrow Q_{h+1}$ and $C(Q) = \{d_1, d_2, \cdots, d_h\}$;

Step 4: Process the case that $|GP(I_{cc})| = 1$, arc s does not intersect arc ω, and $s \neq \upsilon_1$.
(4.1) **if** $\omega \notin B_p(t(\upsilon_1))$, **then goto** Step 5;
(4.2) let $\mathcal{R} = \{r \in F | r \in B_p(h(\omega)) \setminus F(L(Q))\}$;
(4.3) let z be an arc of $\mathcal{R}$ such that $right(I_{cc}(z))$ is the smallest in $I_{cc}(\mathcal{R})$;
(4.4) **if** $|GP(I_{cc} \setminus \{I_{cc}(z)\})| = 1$, **then**
(4.5) let $GP(I_{cc} \setminus \{I_{cc}(z)\}) = \{Q_{\bar{z}}\}$ and let $\omega_{\bar{z}} = F(end(Q_{\bar{z}}))$;
(4.6) **if** arc z intersects arc $\omega_{\bar{z}}$, **then output** "$z \rightarrow Q_{\bar{z}} \rightarrow z$" as a Hamiltonian cycle of $G(F)$ and **terminate** the algorithm.
(4.7) let $F = (F \setminus \{z\}) \cup seg(h(z), h(\omega))$; /* remove $seg(h(\omega), t(z))$ of arc z from F */
(4.8) **goto** Step 4.1;

Step 5: **output** "$G(F)$ has no Hamiltonian cycle" and **terminate** the algorithm.

In the following, we give an example to explain Algorithm HC(F). Let F be the set of arcs shown in Fig. 2. The algorithm picks arc $\mu = a_1$ that does not contain any other arc. It first maps F into the set I_c of intervals by a clockwise mapping w.r.t. $B_a(a_1)$ and a_1 (as shown in Fig. 4). This first greedy path P output by Algorithm GP(I_c) is $a_1 \rightarrow a_2 \rightarrow a_3 \rightarrow a_5 \rightarrow a_4 \rightarrow a_6 \rightarrow a_9 \rightarrow a_7 \rightarrow a_{10} \rightarrow a_8$ which is not a Hamiltonian path of $G(I_c)$. It is straightforward to verify that $\upsilon = F(end(P)) = a_8$. We find that $G(I_c)$ is not Hamiltonian. Next, the algorithm computes $C(P) = L(P) = \{a_9, a_{10}\}$. It is straightforward to verify that $P_1 = a_1 \rightarrow a_2 \rightarrow a_3 \rightarrow a_5 \rightarrow a_4 \rightarrow a_6$ and $\upsilon_1 = F(end(P_1)) = a_6$. The algorithm then maps F into the set I_{cc} of intervals by a counterclockwise mapping w.r.t. $\emptyset$ and $\upsilon_1 = a_6$ (as shown in Fig. 5). By Algorithm GP, we find that $|GP(I_{cc})| = 1$. Let $GP(I_{cc}) = \{Q\}$. Then, $Q = a_4 \rightarrow a_6 \rightarrow a_3 \rightarrow a_5 \rightarrow a_1 \rightarrow a_{11} \rightarrow a_{10} \rightarrow a_8 \rightarrow a_9 \rightarrow a_7 \rightarrow a_2$. Arc $a_4 = F(start(Q))$ does not intersect arc $a_2 = F(end(Q))$. By the Hamiltonian cycle algorithm for interval graphs, we find that $G(I_{cc})$ has a Hamiltonian cycle $a_4 \rightarrow a_6 \rightarrow a_3 \rightarrow a_2 \rightarrow a_1 \rightarrow a_{11} \rightarrow a_{10} \rightarrow a_8 \rightarrow a_9 \rightarrow a_7 \rightarrow a_5 \rightarrow a_4$. It is also a Hamiltonian cycle of $G(F)$.

4 Analysis of Algorithm

We first show how to prove the correctness of Algorithm HC given a set F of sorted arcs. Consider the case of $\mu = \upsilon$. In this case, $G(F)$ and $G(I_{cc})$ are isomorphic [23]. Hence, $G(F)$ is an interval graph and the algorithm is easily verified to be correct. Consider that $\mu \neq \upsilon$. Suppose $\mu \in B_p(t(\upsilon))$. In [23], we showed that P is a Hamiltonian path of $G(I_c)$. Hence, "$\upsilon \rightarrow P$" is a Hamiltonian cycle of $G(F)$ and the algorithm terminates at Step 1.5. Thus, Algorithm HC(F) is correct when $\mu \neq \upsilon$ and $\mu \in B_p(t(\upsilon))$. From now on, we assume that $\mu \neq \upsilon$ and $\mu \notin B_p(t(\upsilon))$. Notice that arcs are open segments,

no arc in F covers the entire circle, and arc μ does not contain any other arc of F.

Apparently, Algorithm HC(F) is correct if it terminates at either Step 1.5, Step 1.6, Step 3.4 or Step 3.5 since $G(I_c)$ and $G(I_{cc})$ are spanning subgraphs of $G(F)$. We observe that $v_1 = v$ if $L(P) = \emptyset$, and $v_1 \in B_{\rm a}(s)$ if $s \neq v_1$. To verify the correctness of Step 4.6, we show that arc z intersects both $F(start(Q_{\bar{z}}))$ and $F(end(Q_{\bar{z}}))$, and, hence, "$z \rightarrow Q_{\bar{z}} \rightarrow z$" is a Hamiltonian cycle of $G(F)$. To verify the correctness of Steps 3.1 and 5, we find a subset $\mathcal{C}$ of F such that $G(F \setminus \mathcal{C})$ has at least $|\mathcal{C}| + 1$ connected components. By Proposition 2.1, $G(F)$ is not Hamiltonian. In Step 4.7, we remove the segment $seg(h(\omega), t(z))$ of arc z from F. We prove its correctness by finding a subset $\mathcal{C}$ of F such that $z \in \mathcal{C}$, $G(F \setminus \mathcal{C})$ has at least $|\mathcal{C}|$ connected components, and no arcs of $F \setminus \mathcal{C}$ intersect $seg(h(\omega), t(z))$. By Proposition 2.2, $G(F)$ is Hamiltonian if and only if graph $G((F \setminus \{z\}) \cup seg(h(z), h(\omega)))$ is Hamiltonian. To verify the correctness of the algorithm, we may prove the following claims. Due to the space limitation, we omit the proofs of these claims.

(1) $G(F)$ is not Hamiltonian if $|GP(I_{cc})| > 1$ (Step 3.1);
(2) $G(F)$ is not Hamiltonian if $|GP(I_{cc})| = 1$ and $\omega \notin B_{\rm p}(t(v_1))$ (Step 5);
(3) Suppose that $\omega \in B_{\rm p}(t(v_1))$ (it implies $\mathcal{R} \neq \emptyset$) and $|GP(I_{cc} \setminus \{I_{cc}(z)\})| = 1$. Then, $G(F)$ is Hamiltonian if and only if arc z intersects arc $\omega_{\bar{z}}$ in $G(F)$ (Step 4.6);
(4) Suppose that $\omega \in B_{\rm p}(t(v_1))$ and $|GP(I_{cc} \setminus \{I_{cc}(z)\})| = 1$. If arcs z and $\omega_{\bar{z}}$ do not intersect, then $G(F)$ is Hamiltonian if and only if $G((F \setminus \{z\}) \cup seg(h(z), h(\omega)))$ is Hamiltonian (Step 4.7).
(5) Suppose that $\omega \in B_{\rm p}(t(v_1))$ and $|GP(I_{cc} \setminus \{I_{cc}(z)\})| > 1$. Then, $G(F)$ is Hamiltonian if and only if $G((F \setminus \{z\}) \cup seg(h(z), h(\omega)))$ is Hamiltonian (Step 4.7).

Next, we analyze the complexity of Algorithm HC. The algorithm is assumed that the input graph is given by a circular-arc model F that is a set of n sorted arcs. Given such a model F, an arc μ of F that does not contain any other arc can be easily found in $O(n)$ time. From now on, we assume that such an arc μ is given.

Manacher et al. proposed a Hamiltonian cycle algorithm on interval graphs [30]. Chang et al. implemented the algorithm in $O(n)$ time if a set of n sorted intervals is given [7]. Hence, testing whether $G(I_c)$ or $G(I_{cc})$ has a Hamiltonian cycle can be done in $O(n)$ time. It is easy to see that the algorithm runs in $O(n)$ time if it terminates at either Step 1.5, Step 3.1 or Step 3.4. Therefore, the time complexity of Algorithm HC is $O(n)$ if it terminates before Step 4. Consider that Algorithm HC does not terminate before Step 4. Let Δ_ω be the number of arcs in $B_{\rm p}(h(\omega))$. Clearly, $\Delta_\omega \leq \Delta$. Let $\mathcal{R} = \{r \in F | r \in B_{\rm p}(h(\omega)) \setminus F(L(Q))\}$. In each iteration of Step 4, $|\mathcal{R}|$ is decreased by 1. Obviously, $|\mathcal{R}| \leq \Delta_\omega$. Therefore, Step 4 is iterated at most $|\mathcal{R}|$ times, and, hence, it is iterated at most Δ_ω times. Since every line can be implemented in $O(n)$ time, Step 4 runs in $O(\Delta n)$ time. Thus, we conclude the following theorem.

Theorem 4.1. *Given a circular-arc model F that is a set of n sorted arcs, Algorithm HC solves the Hamiltonian cycle problem on $G(F)$ in $O(\Delta n)$ time.*

In [23], we proposed an $O(n)$-time approximation algorithm for the path cover problem on a circular-arc graph $G(F)$. Let π' be the cardinality of the path cover found by the approximation algorithm. We showed that $\pi' \leq \pi(G(F)) + 1$ in [23]. Then, the path cover problem on $G(F)$ can be solved by reducing it to the Hamiltonian cycle problem as follows. If $\pi' = 1$, then $\pi(G(F)) = \pi'$. Otherwise, $\pi(G(F)) = \pi' - 1$ if and only if $G(F) \otimes K_{\pi'-1}$ is Hamiltonian, where $K_{\pi'-1}$ is a complete graph of $\pi' - 1$ vertices, $G(F)$ and $K_{\pi'-1}$ are disjoint, and $G(F) \otimes K_{\pi'-1}$ is the graph obtained by connecting every vertex of $G(F)$ with all vertices of $K_{\pi'-1}$. Apparently, $G(F) \otimes K_{\pi'-1}$ is a circular-arc graph, too. Let Δ be the maximum degree of $G(F)$. Then, the maximum degree of $G(F) \otimes K_{\pi'-1}$ is $\Delta + (\pi' - 1)$. By using the above reduction and the Hamiltonian cycle algorithm proposed in the paper, the path cover problem on $G(F)$ can be solved in $O(n^2)$ time. Thus, we have the following corollary.

Corollary 4.2. *Given a circular-arc model F that is a set of n sorted arcs, the path cover problem on $G(F)$ can be solved in $O(n^2)$ time.*

It is well known that the Hamiltonian cycle and Hamiltonian path problems on circular-arc graphs can be reduced to each other [23, 27]. Thus, the Hamiltonian path problem on circular-arc graphs can be solved by an algorithm whose time complexity is the same as that of the most efficient algorithm for the Hamiltonian cycle problem on the same class of graphs. Therefore, the following corollary immediately holds true.

Corollary 4.3. *Given a circular-arc model F that is a set of n sorted arcs, the Hamiltonian path problem on $G(F)$ can be solved in $O(\Delta n)$ time.*

5 Concluding Remarks

In this paper, we solve the Hamiltonian cycle problem on circular-arc graphs in $O(\Delta n)$ time. This improves the best previous result in [36] for this problem which is an $O(n^2 \log n)$-time algorithm. Using a general reduction technique in [23] and [27], the Hamiltonian path problem on circular-arc graphs can be solved in $O(\Delta n)$ time, too. On the other hand, using the reduction technique in [23] and the Hamiltonian cycle algorithm presented in the

paper, the path cover problem on circular-arc graphs can be solved in $O(n^2)$ time. In addition, given a circular-arc model F that is a set of n arcs with endpoints sorted, whether the Hamiltonian cycle or Hamiltonian path problem on $G(F)$ can be solved in $O(n)$ time remains open.

References

[1] S.R. Arikati and C. Pandu Rangan, Linear algorithm for optimal path cover problem on interval graphs, *Inform. Process. Lett.* **35** (1990) 149–153.

[2] N. Ascheuer, Hamiltonian path problems in the online optimization of flexible manufacturing systems, Technique Report TR 96-3, Konrad-Zuse-Zentrum für Informationstechnik, Berlin, 1996.

[3] J.C. Bermond, Hamiltonian graphs, in Selected Topics in Graph Theory ed. by L.W. Beinke and R.J. Wilson, Academic Press, New York, 1978.

[4] A.A. Bertossi and M.A. Bonuccelli, Hamiltonian circuits in interval graph generalizations, *Inform. Process. Lett.* **23** (1986) 195–200.

[5] M.A. Bonuccelli and D.P. Bovet, Minimum node disjoint path covering for circular-arc graphs, *Inform. Process. Lett.* **8** (1979) 159–161.

[6] M.S. Chang, S.L. Peng and J.L. Liaw, Deferred-query – an efficient approach for some problems on interval and circular-arc graphs, Lecture Notes in Comput. Sci., vol. 709, Springer, Berlin, 1993, pp. 222–233.

[7] M.S. Chang, S.L. Peng and J.L. Liaw, Deferred-query: an efficient approach for some problems on interval graphs, *Networks* **34** (1999) 1–10.

[8] D.Z. Chen, D.T. Lee, R. Sridhar and C.N. Sekharan, Solving the all-pair shortest path query problem on interval and circular-arc graphs, *Networks* **31** (1998) 249–258.

[9] P. Damaschke, The Hamiltonian circuit problem for circle graphs is NP-complete, *Inform. Process. Lett.* **32** (1989) 1–2.

[10] P. Damaschke, J.S. Deogun, D. Kratsch and G. Steiner, Finding Hamiltonian paths in cocomparability graphs using the bump number algorithm, *Order* **8** (1992) 383–391.

[11] P. Damaschke, Paths in interval graphs and circular arc graphs, *Discrete Math.* **112** (1993) 49–64.

[12] J.S. Deogun and G. Steiner, Polynomial algorithms for Hamiltonian cycle in cocomparability graphs, *SIAM J. Comput.* **23** (1994) 520–552.

[13] J.S. Deogun and C. Riedesel, Hamiltonian cycles in permutation graphs, *J. Combin. Math. Combin. Comput.* **27** (1998) 161–200.

[14] E.M. Eschen and J.P. Spinrad, An $O(n^2)$ algorithm for circular-arc graph recognition, in: Proceedings of the 4th Annual ACM-SIAM Symposium on Discrete Algorithm (SODA'93), 1993, pp. 128–137.

[15] M.R. Garey and D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, Freeman, San Francisco, CA, 1979.

[16] M.C. Golumbic, Algorithmic Graph Theory and Perfect Graphs, Academic Press, New York, 1980.

[17] S.Y. Hsieh, C.W. Ho, T.S. Hsu and M.T. Ko, The Hamiltonian problem on distance-hereditary graphs, *Discrete Appl. Math.* **153** (2006) 508–524.

[18] W.L. Hsu and K.H. Tsai, Linear time algorithms on circular-arc graphs, *Inform. Process. Lett.* **40** (1991) 123–129.

[19] W.L. Hsu, $O(M \cdot N)$ algorithms for the recognition and isomorphism problems on circular-arc graphs, *SIAM J. Comput.* **24** (1995) 411–439.

[20] R.W. Hung, S.C. Wu and M.S. Chang, Hamiltonian cycle problem on distance-hereditary graphs, *J. Inform. Sci. Eng.* **19** (2003) 827–838.

[21] R.W. Hung and M.S. Chang, A simple linear algorithm for the connected domination problem in circular-arc graphs, *Discuss. Math. Graph Theory* **24** (2004) 137–145.

[22] R.W. Hung and M.S. Chang, Linear-time algorithms for the Hamiltonian problems on distance-hereditary graphs, *Theoret. Comput. Sci.* **341** (2005) 411–440.

[23] R.W. Hung and M.S. Chang, Solving the path cover problem on circular-arc graphs by using an approximation algorithm, *Discrete Appl. Math.* **154** (2006) 76–105.

[24] A. Itai, C.H. Papadimitriou and J.L. Szwarcfiter, Hamiltonian paths in grid graphs, *SIAM J. Comput.* **11** (1982) 676–686.

[25] D.S. Johnson, The NP-complete column: an ongoing guide, *J. Algorithms* **6** (1985) 434–451.

[26] M.S. Krishnamoorthy, An NP-hard problem in bipartite graphs, *SIGACT News* **7** (1976) 26.

[27] J. van Leeuwen, Graph Algorithms: Handbook of Theoretical Computer Science vol. A, Elsevier, Amsterdam, 1990.

[28] Y.D. Liang and G.K. Manacher, An $O(n \log n)$ algorithm for finding a minimal path cover in circular-arc graphs, in: Proceedings of the ACM conference on Computer Science, Indianapolis, IN, USA, 1993, pp. 390–397.

[29] Y.D. Liang, G.K. Manacher, C. Rhee and T.A. Mankus, A linear algorithm for finding Hamiltonian circuits in circular-arc graphs, in: Proceedings of the 32nd Southeast ACM conference, 1994, pp. 15–22.

[30] G.K. Manacher, T.A. Mankus and C.J. Smith, An optimum $\Theta(n \log n)$ algorithm for finding a canonical Hamiltonian path and a canonical Hamiltonian circuit in a set of intervals, *Inform. Process. Lett.* **35** (1990) 205–211.

[31] S. Masuda and K. Nakajima, An optimal algorithm for finding a maximum independent set of a circular-arc graph, *SIAM J. Comput.* **17** (1988) 41–52.

[32] R.M. McConnell, Linear-time recognition of circular-arc graphs, *Algorithmica* **37** (2003) 93–147.

[33] J.F. O'Callaghan, Computing the perceptual boundaries of dot patterns, *Comput. Graphics Image Process.* **3** (1974) 141–162.

[34] F.P. Preperata and M.I. Shamos, Computational Geometry: An Introduction, Springer, New York, 1985.

[35] C. Riedesel and J.S. Deogun, Permutation graphs: Hamiltonian paths, *J. Combin. Math. Combin. Comput.* **19** (1995) 55–63.

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

[37] G.T. Toussaint, Pattern recognition and geometrical complexity, in: Proceedings of the 5th International Conference on Pattern Recognition, Miami Beach, 1980, pp. 1324–1347.

[38] A.C. Tucker, An efficient test for circular-arc graphs, *SIAM J. Comput.* **9** (1980) 1–24.

[39] M.S. Waterman and J.R. Griggs, Interval graphs and maps of DNA, *Bull. Math. Biol.* **48** (1986) 189–195.