An Optimal Algorithm for Finding Locally Connected Spanning Trees on Circular-Arc Graphs

Ching-Chi Lin‡ Gerard J. Chang†* Gen-Huey Chen‡

‡Department of Computer Science and Information Engineering
National Taiwan University, Taipei, Taiwan
Email:{d91018,ghchen}@csie.ntu.edu.tw

†Department of Mathematics
National Taiwan University, Taipei, Taiwan
Email: gjchang@math.ntu.edu.tw

Abstract

Suppose that T is a spanning tree of a graph G. T is called a locally connected spanning tree of G if for every vertex of T, the set of all its neighbors in T induces a connected subgraph of G. In this paper, given an intersection model of a circular-arc graph, an $O(n)$-time algorithm is proposed that can determine whether the circular-arc graph contains a locally connected spanning tree or not, and produce one if it exists.

1 Introduction

A communication network is conveniently represented with a graph G. The vertex set of G, denoted by $V(G)$, represents the set of nodes in the network, and the edge set of G, denoted by $E(G)$, represents the set of communication links. In this paper, we use xy to represent the edge connecting vertices x and y. When a data packet is required to be transmitted from a node to another node in a communication network, it will be carried through a path that consists of many communication links. Thus, it is cost effective to build a communication network as a tree network. However, such a tree network is fragile to any fault, node fault or link fault, because its connectivity is only one. In order to enhance the fault tolerance, Farley [8, 9] introduced the concept of *isolated failure immune* (IFI) networks.

A set of node failures (i.e., node faults) is *isolated* if every two of them are not adjacent. A connected network is *immune* to a set of node failures if it remains connected after removing these node failures. An IFI network is immune to any isolated set of node failures. In [2], Cai suggested an instance of IFI networks. Let T be a spanning tree of G and $N_T(v)$ be the set of all neighboring vertices of v in T. If for every $v \in V(G)$, the subgraph of G induced by $N_T(v)$ is connected, then T is called a *locally connected spanning tree* (LCST) of G. Figure 1 shows two spanning trees, T_1 and T_2, of a graph G, where T_1 is an LCST. Since the subgraph of G induced by $N_{T_2}(u)$ (and $N_{T_2}(v)$) is not connected, T_2 is not an LCST. A network containing an LCST is an IFI network.

In [3], the problem of determining whether a planar graph or a split graph contains an LCST was shown to be NP-complete. Moreover, two algorithms, requiring $O(|V(G)|+|E(G)|)$ time, were proposed to find an LCST in a 2-connected directed path graph [6] and to produce an LCST from a spanning tree of a given graph by augment-

*Supported in part by the National Science Council under grant NSC-95-2221-E-002-125-MY3. Taida Institute for Mathematical Sciences, National Taiwan University, Taipei 10617, Taiwan. National Center for Theoretical Sciences, Taipei Office.

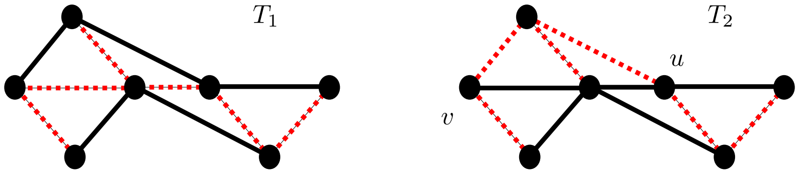

Figure 1: Two spanning trees (dotted edges) of G.

ing fewest edges, respectively. In [17], the authors presented two algorithms to find an LCST in a 2-connected strongly chordal graph [4, 7, 20] and an LCST in a proper circular-arc graph, respectively, also in $O(|V(G)| + |E(G)|)$ time.

Circular-arc graphs, which are a superfamily of proper circular-arc graphs, are a natural generalization of interval graphs. A lot of optimization problems, e.g., the maximum independent set problem [11–14, 16, 18], the minimum clique cover problem [12, 14], the minimum cut problem [16, 21], and the minimum dominating set problem [5, 14, 15], have been studied on circular-arc graphs. These problems are all NP-complete if they are defined on general graphs, and solvable in $O(|V(G)|+|E(G)|)$ time if they are defined on circular-arc graphs. So, it is interesting to investigate whether the LCST problem is NP-complete or polynomial time solvable when it is defined on circular-arc graphs.

In this paper, we show that the LCST problem on a circular-arc graph G is polynomial time solvable. To say more concretely, given an intersection model F of G, an $O(|V(G)|)$ time algorithm is proposed that can determine whether G contains an LCST or not, and produce it if it exists.

2 Preliminaries

$G = (V(G), E(G))$ is a *circular-arc graph* [10, 19, 22] if there is a one-to-one correspondence between $V(G)$ and a set of arcs so that $(u, v) \in E(G)$ if and only if the corresponding arc of u overlaps with the corresponding arc of v. In the rest of this paper, we let $n = |V(G)|$ and $m = |E(G)|$. McConnell [19] gave an $O(n + m)$-time algorithm to recognize a circular-arc graph G, and as a byproduct, an intersection model of G can be obtained simultaneously. In the rest of this paper, we denote the intersection model by F, and assume that F is available to G.

For each $v \in V(G)$, let $a(v)$ denote the corresponding arc of v in F. Further, for any subset W of $V(G)$, we define $a(W) = \{a(v)|\ v \in W\}$. Each arc is represented with $[h(v), t(v)]$, where $h(v)$ is the *head* of $a(v)$, $t(v)$ is the *tail* of $a(v)$, and $h(v)$ precedes $t(v)$ in a counterclockwise traversal. We assume that all arc endpoints (i.e., $h(v)$ and $t(v)$) are distinct and no arc covers the entire circle.

Let $d(v)$ denote the *density* of $a(v)$, which is the number of arcs (including $a(v)$) in F that contain $h(v)$. A *segment* of a circle is a continuous part between two endpoints. We use $[s, t]$ to denote a segment from endpoint s to endpoint t in a counterclockwise traversal. Similarly, we use (s, t) ($(s, t]$ and $[s, t)$, respectively) to denote the same segment, but excluding s and t (s and t, respectively).

A subset S of $V(G)$ is a *separating set* of G if the subgraph of G induced by $V(G) - S$ contains more than one connected component (component for short). When $S = \{v\}$, v is called a *cut vertex* of G. If G contains no separating set of size smaller than k, then G is *k-connected*. In subsequent discussion, we use $G[S]$ to denote the subgraph of G induced by a subset S of $V(G)$.

Lemma 1 ([3]) *If G has an LCST, then G is 2-connected and for every separating set S of G,*

$G[S]$ contains at least one edge of the LCST.

Lemma 2 ([17]) *If G is a circular-arc graph with $d(v) \leq 2$ for four or more distinct vertices v, then G has no LCST.*

Lemma 3 ([17]) *If there is a separating set $\{x, y\}$ of G and a component H of $G - \{x, y\}$ so that H contains no common neighbor of x and y, then G has no LCST.*

We use $G_1 \cup G_2$ to denote the *union* of two graphs G_1 and G_2, which is the graph with vertex set $V(G_1) \cup V(G_2)$ and edge set $E(G_1) \cup E(G_2)$.

Lemma 4 *Suppose that T_1 is an LCST of G_1 and T_2 is an LCST of G_2. If $V(T_1) \cap V(T_2) = \{x, y\}$ and $E(T_1) \cap E(T_2) = \{xy\}$, then $T_1 \cup T_2$ is an LCST of $G_1 \cup G_2$.*

Proof. Let $T = T_1 \cup T_2$ and $G = G_1 \cup G_2$. It suffices to show that both $G[N_T(x)]$ and $G[N_T(y)]$ are connected. Since $G_1[N_{T_1}(x)]$ and $G_2[N_{T_2}(x)]$ are connected and $y \in N_{T_1}(x) \cap N_{T_2}(x)$, $G[N_T(x)]$ is connected. Similarly, $G[N_T(y)]$ is connected. □

In the rest of this section, we let G be an interval graph [1] with $V(G) = \{v_1, v_2, \ldots, v_n\}$, where $n \geq 3$. Since an interval graph is also a circular-arc graph, we use $a(v)$ to denote the corresponding interval of v. It is assumed that the left endpoint of $a(v_i)$ is on the left of the left endpoint of $a(v_{i+1})$, where $1 \leq i \leq n-1$. In [17], the authors presented an $O(n+m)$ time algorithm, i.e., Algorithm Strongly-Chordal, that can construct an LCST in a 2-connected strongly chordal graph.

Algorithm Strongly-Chordal selects an incident edge for each vertex so that the collection of all selected edges forms an LCST. However, with F, $O(n)$ time is sufficient to construct an LCST in an interval graph G, as explained below. Suppose that $v_i v_{i^*}$ is the edge to be selected for vertex v_i. We set $v_{1^*} = v_2$ and for $2 \leq i \leq n$, determine v_{i^*} so that $a(v_{i^*})$ has the rightmost right endpoint among $a(W)$, where $W = \{v_k | \ v_k \in N_G(v_i)$ and $k < i\}$. Since v_{i^*} is v_{i-1} or $v_{(i-1)^*}$, all edges $v_i v_{i^*}$ can be determined in $O(n)$ time. The collection of all edges $v_i v_{i^*}$ forms an LCST of G, with the same arguments as Algorithm Strongly-Chordal. Therefore, the following lemma is immediate.

Lemma 5 *Suppose that G is a 2-connected interval graph. Then, an LCST containing $v_1 v_2$ can be obtained in $O(n)$ time.*

Suppose $v_x \in N_G(v_1)$. The following algorithm can find an LCST of G, if it exists, that contains $v_1 v_x$.

Algorithm LCST-Interval.

(1) If G contains a cut vertex or G contains a vertex v_s such that $\{v_1, v_s\}$ and $\{v_x, v_s\}$ are two separating sets of G, then stop. /* No LCST exists in G. */

(2) Let $h = max\{i| \ v_1 v_i \in E(G)\}$. If $h = n$, then let $T = \{v_1v_2, v_1v_3, \ldots, v_1v_n\}$ and perform step (8).

(3) Let $W = \{v_k | \ v_k \in N_G(v_1) - \{v_x\}$ and $\{v_1, v_k\}$ is not a separating set of $G\}$. Find the vertex v_p of W so that $a(v_p)$ has the rightmost right endpoint among $a(W)$.

(4) Let $W' = \{v_l | \ v_l \in N_G(v_1) - \{v_p\}\}$. Find the vertex v_q of W' so that $a(v_q)$ has the rightmost right endpoint among $a(W')$.

(5) Let $T_1 = \{v_1v_2, \ldots, v_1v_{p-1}, v_1v_{p+1}, \ldots, v_1v_h\} \cup \{v_pv_q\}$.

(6) Construct an LCST T_2 in $G[\{v_p, v_q, v_{h+1}, v_{h+2}, \ldots, v_n\}]$ with $v_pv_q \in E(T_2)$.

(7) Let $T = T_1 \cup T_2$.

(8) Output T. /* T is an LCST of G that contains v_1v_x. */

In the following discussion, we let $G_1 = G[\{v_1, v_2, \ldots, v_h\}]$ and $G_2 = G[\{v_p, v_q\} \cup \{v_{h+1}, v_{h+2}, \ldots, v_n\}]$. Notice that $V(G_1) \cap V(G_2) = \{v_p, v_q\}$.

Lemma 6 *Suppose that G is 2-connected and $h < n$. If G contains no vertex v_s, then G_2 is 2-connected.*

Proof. Let v_α, v_β and v_γ denote the three vertices in $N_G(v_1)$ so that $a(v_\alpha), a(v_\beta)$ and $a(v_\gamma)$ have the rightmost, the second rightmost and the third rightmost right endpoints, respectively, among

$a(N_G(v_1))$. Suppose conversely that G contains no vertex v_s and G_2 is not 2-connected. If $\{v_p, v_q\} = \{v_\alpha, v_\beta\}$, then G_2 is 2-connected, for otherwise G is not 2-connected, a contradiction. This contradicts to our assumption that G_2 is not 2-connected. Therefore, we have $\{v_p, v_q\} \neq \{v_\alpha, v_\beta\}$.

As a consequence of step (4), we have $v_q \in \{v_\alpha, v_\beta\}$, which implies $v_p \notin \{v_\alpha, v_\beta\}$. Again, as a consequence of step (3), both $\{v_1, v_\alpha\}$ and $\{v_1, v_\beta\}$ are separating sets of G, or one of v_α and v_β is v_x and the other together with v_1 forms a separating set of G. Notice that the former is not true unless G is not 2-connected. Therefore, the latter holds, i.e., $v_\alpha = v_x$ or $v_\beta = v_x$. Without loss of generality, we assume $v_\alpha = v_x$ (and hence $\{v_1, v_\beta\}$ is a separating set of G).

Let $r_G(v_i)$ be the number of intervals in $a(N_G(v_i))$ which contain the right endpoint of $a(v_i)$. Since G is 2-connected and $h < n$, we have $r_G(v_1) \geq 2$. If $r_G(v_1) = 2$, then $a(v_\alpha)$ and $a(v_\beta)$ are the two intervals that contain the right endpoint of $a(v_1)$, i.e., $\{v_\alpha, v_\beta\}$ is a separating sets of G. However, this contradicts to our assumption about v_s, because $v_\alpha = v_x$ and $\{v_1, v_\beta\}$ is a separating set of G (i.e., $v_s = v_\beta$).

On the other hand, if $r_G(v_1) > 2$, then $a(v_\alpha)$, $a(v_\beta)$ and $a(v_\gamma)$ exist. And, $a(v_\alpha)$, $a(v_\beta)$ and $a(v_\gamma)$ contain the right endpoint of $a(v_1)$. Notice that both $\{v_1, v_\beta\}$ and $\{v_1, v_\gamma\}$ are not separating sets of G similarly, which implies $\{v_1, v_\gamma\}$ is not a separating set of G. Hence, we have $v_p = v_\gamma$ as a consequence of step (3), and $v_q = v_\alpha$ as a consequence of step (4). Now that G_2 is not 2-connected, there exists a vertex $v_t \in V(G_2)$ with $r_{G_2}(v_t) < 2$.

Recall that G_2 is 2-connected provided $\{v_p, v_q\} = \{v_\alpha, v_\beta\}$. It implies $r_{G[V(G_2)\cup\{v_\beta\}]}(v_t) \geq 2$, i.e., the right endpoint of $a(v_t)$ is contained in $a(v_\beta)$ (and hence in $a(v_\alpha)$). Since $r_{G_2}(v_t) < 2$, the right endpoint of $a(v_t)$ is not contained in $a(v_\gamma)$. Hence, $r_G(v_t) = 2$, which implies that $\{v_\alpha, v_\beta\}$ is a separating set of G. Similarly, there is a contradiction to our assumption about v_s. □

Lemma 7 *There is an LCST of G that contains v_1v_x if and only if Algorithm LCST-Interval outputs T. Moreover, T is such an LCST, which can be obtained in $O(n)$ time.*

Proof. According to Lemma 1, G has no LCST if G has a cut vertex, and G has no LCST containing v_1v_x if both $\{v_1, v_s\}$ and $\{v_x, v_s\}$ are separating sets of G. Therefore, we need only to consider the situation that G is 2-connected and no v_s exists in G. When $h = n$, $T = \{v_1v_2, v_1v_3, \ldots, v_1v_n\}$ is clearly an LCST of G that contains v_1v_x. In subsequent discussion, $h < n$ is assumed.

Since $v_p \neq v_x$, we have $v_1v_x \in E(T_1)$. Clearly, $V(G_1) \cup V(G_2) = V(G)$ and $E(G_1) \cup E(G_2) \subseteq E(G)$. If $T = T_1 \cup T_2$ is an LCST of $G_1 \cup G_2$, then T is an LCST of G as well. Since $V(T_1) \cap V(T_2) = \{v_p, v_q\}$ and $E(T_1) \cap E(T_2) = \{v_pv_q\}$, by Lemma 4 we only need to show that T_1 is an LCST of G_1 and to construct an LCST, i.e., T_2, of G_2 with $v_pv_q \in E(T_2)$ below.

In order to show that T_1 is an LCST of G_1, it suffices to show that $v_pv_q \in E(G_1)$ and both $G_1[N_{T_1}(v_1)]$ and $G_1[N_{T_1}(v_q)]$ are connected. By Lemma 6, G_2 is 2-connected, which implies that both $a(v_p)$ and $a(v_q)$ contain the left endpoint of $a(v_{h+1})$, i.e., $v_pv_q \in E(G)$. Since $\{v_1, v_p\}$ is not a separating set of G, $G[\{v_2, \ldots, v_{p-1}, v_{p+1}, \ldots, v_n\}]$ is connected, which further implies that $G[\{v_2, \ldots, v_{p-1}, v_{p+1}, \ldots, v_h\}] = G_1[N_{T_1}(v_1)]$ is connected. Since $v_p \in N_G(v_1)$ and $N_{T_1}(v_q) = \{v_1, v_p\}$, we know that $G_1[N_{T_1}(v_q)]$ is connected. On the other hand, according to Lemma 5 and Lemma 6, T_2 can be obtained in $O(n)$ time.

Next we show that Algorithm LCST-Interval runs in $O(n)$ time. With F, step (1) can be completed in $O(n)$ time. The vertex v_s can be obtained by taking the intersection of the two sets $\{v_c |\ \{v_1, v_c\}$ is a separating set of $G\}$ and $\{v_d |\ \{v_x, v_d\}$ is a separating set of $G\}$, which requires $O(n)$ time by the aid of F. The other steps can be completed also in $O(n)$ time. □

Since Algorithm LCST-Interval outputs T if and only if the if-condition of step (1) is not satisfied, Lemma 7 can be rewritten as follows.

Lemma 8 *There is an LCST of G that contains v_1v_x if and only if G is 2-connected and there is no vertex v_s in G such that $\{v_1, v_s\}$ and $\{v_x, v_s\}$ are two separating sets of G. Moreover, the LCST can be obtained in $O(n)$ time.*

If a circular-arc graph G has $d(v) = 1$ for some vertex $v \in V(G)$, then G is also an interval graph. Hence, an LCST of G can be found, if it exists, according to the work of [3]. Moreover, according to Lemma 2, G has no LCST provided G has $d(v) = 2$ for four or more distinct vertices v. In Sections 3, an $O(n)$-time algorithm is proposed for the situation when no vertex v with $d(v) = 2$. The algorithm can determine whether G contains an LCST or not, and produce one if it exists. We omit the three situations: exactly three vertices v, exactly two vertices v and exactly one vertex v with $d(v) = 2$ due to the limitation in the number of pages. We suppose $V(G) = \{v_1, v_2, \ldots, v_n\}$, where $n \geq 3$ in the following discussion.

3 No vertex v with $d(v) = 2$

Suppose that G has no vertex v with $d(v) = 2$. An algorithm is proposed in this section, which can produce an LCST of G, if it exists. To begin with, the algorithm finds an ordering $v_{p(1)}, v_{p(2)}, \ldots, v_{p(n)}$ of vertices in order to construct an LCST of G. Define $S_q = \{v_h | \ a(v_h) \text{ contains } q\}$, where q is a point of the circle in F, and $K_{x,y} = \{v_k | \ v_k \in N_G(v_x) - \{v_y\},\ \{v_x, v_k\}$ is a separating set of G, and there exists one component C of $G - \{v_x, v_k\}$ so that all arcs of $a(V(C))$ are contained in the segment $(h(v_x), h(v_y))$ in $F\}$. The selection of $v_{p(1)}$ and $v_{p(2)}$ requires that $a(v_{p(1)}) \cap a(v_{p(2)})$ is not empty and satisfies the following two conditions.

(C1) There exists a point q of the circle in F so that $t(v_{p(1)})$ and $t(v_{p(2)})$ are the last two tails encountered among all the corresponding tails of S_q in F if a counterclockwise traversal from q is made.

(C2) $K_{p(1),p(2)}$ is empty.

Then, $v_{p(3)}, v_{p(4)}, \ldots, v_{p(n)}$ are determined so that $h(v_{p(i+1)})$ immediately succeeds $h(v_{p(i)})$ in a counterclockwise traversal, where $2 \leq i \leq n-1$. There is an LCST of G if and only if such an ordering can be found.

Suppose that H is a subgraph of G. For each $v_l \in V(H)$, define $\tilde{N}_H(v_l) = \{v_k | \ v_k \in N_H(v_l)$ and $a(v_k)$ contains $h(v_l)\}$, i.e., $\tilde{N}_H(v_l)$ is the set of neighbors of v_l in H whose corresponding arcs contain $h(v_l)$ in F. Also let $\tilde{v}_{l,H}$ denote the vertex of $\tilde{N}_H(v_l)$ whose corresponding tail in F is encountered last among all vertices of $\tilde{N}_H(v_l)$ if a counterclockwise traversal from $h(v_l)$ is made. Let $G_{p(i)} = G[\{v_{p(1)}, v_{p(2)}, \ldots, v_{p(i)}\}]$, where $1 \leq i \leq n$. Figure 2 shows an example, where $\tilde{N}_{G_{p(4)}}(v_{p(4)}) = \{v_{p(1)}, v_{p(2)}\}$ and $\tilde{v}_{p(4),G_{p(4)}} = v_{p(2)}$.

The following is a formal description of the algorithm.

Algorithm LCST-Circular-Arc-0.

(1) Arbitrarily select a point q of the circle in F and determine two vertices v_x and v_y from S_q so that $t(v_x)$ and $t(v_y)$ are the last two tails encountered among all the corresponding tails of S_q in F if a counterclockwise traversal from q is made. Without loss of generality, suppose that $a(v_x)$ contains $h(v_y)$.

(2) Set $s_0 = x$, $s_1 = y$, $m_0 = x$, and $m_1 = y$.

(3) Repeat

If K_{m_0,m_1} is not empty, then

(3.1) Determine $v_d \in K_{m_0,m_1}$ so that $a(v_d)$ is the last arc encountered among all the corresponding arcs of K_{m_0,m_1} in F if a clockwise traversal from $h(v_{m_1})$ is made.

(3.2) If $a(v_d)$ contains $h(v_{m_0})$, then set $(m_0, m_1) = (d, m_0)$. Otherwise, set $m_1 = d$.

Until K_{m_0,m_1} is empty or $(m_0, m_1) = (s_0, s_1)$.

(4) If $(m_0, m_1) = (s_0, s_1)$, then stop. /* No LCST exists in G. */

(5) Determine $v_{p(1)} = v_{m_0}, v_{p(2)} = v_{m_1}$, and $v_{p(3)}, v_{p(4)}, \ldots, v_{p(n)}$ so that $h(v_{p(i+1)})$ immediately succeeds $h(v_{p(i)})$ in a counterclockwise traversal, where $2 \leq i \leq n-1$.

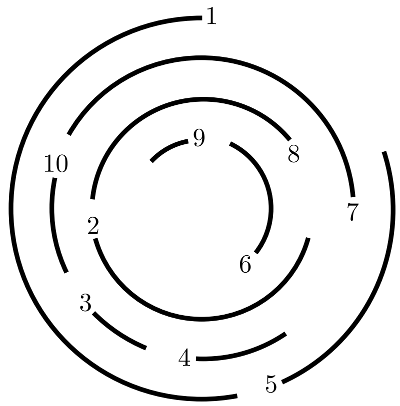

Figure 2: An example with $\tilde{N}_{G_{p(4)}}(v_{p(4)}) = \{v_{p(1)}, v_{p(2)}\}$ and $\tilde{v}_{p(4),G_{p(4)}} = v_{p(2)}$.

(6) Set $T^{(2)} = \{v_{p(1)}v_{p(2)}\}$.

(7) For $i = 3$ to n, perform the following steps.

(7.1) If $|\tilde{N}_{G_{p(i)}}(v_{p(i)})| = 2$ and $h(v_{p(i)}) \in (h(v_{p(1)}), h(v_{p(2)}))$, set $T^{(n)} = T^{(i-1)} \cup \{v_{p(1)}v_{p(i)}, v_{p(1)}v_{p(i+1)}, \ldots, v_{p(1)}v_{p(n)}\}$ and go to step (8).

(7.2) Set $T^{(i)} = T^{(i-1)} \cup \{v_{p(i)}\tilde{v}_{p(i),H_{p(i)}}\}$, where $H_{p(i)} = G_{p(i)} - \{v_{p(1)}\}$ if $h(v_{p(i)}) \in (h(v_{p(1)}), h(v_{p(2)}))$ and $H_{p(i)} = G_{p(i)}$ else.

(8) Output $T^{(n)}$.

Steps (1) to (3) try to find $v_{p(1)}$ and $v_{p(2)}$, i.e., $p(1) = m_0$ and $p(2) = m_1$ if v_{m_0} and v_{m_1} satisfy (C1) and (C2). Step (3) starts with $(v_{m_0}, v_{m_1}) = (v_{s_0}, v_{s_1})$ and traverses the circle clockwise until finding a feasible pair of v_{m_0} and v_{m_1} or returning to (v_{s_0}, v_{s_1}). There is no LCST in G for the latter case. If the current pair of v_{m_0} and v_{m_1} do not satisfy (C2), then the next pair of v_{m_0} and v_{m_1} are determined according to steps (3.1) and (3.2). Notice that the first pair of v_{m_0} and v_{m_1} satisfy (C1), and each subsequent pair of v_{m_0} and v_{m_1} also satisfy (C1), as explained below.

Refer to Figure 3 for an illustrative example, where $v_{m'_0}$ and $v_{m'_1}$ denote the next pair of v_{m_0} and v_{m_1}. We have $K_{m_0,m_1} = \{v_{k_1}, v_{k_2}\}$ and $v_d = v_{k_2}$. Notice that $\{v_{c_1}\}(\{v_{c_2}\})$ is one component of $G - \{v_{m_0}, v_{k_1}\}(G - \{v_{m_0}, v_{k_2}\})$. If $a(v_d)$ contains $h(v_{m_0})$, shift v_{m_0} to v_d and v_{m_1} to v_{m_0}, i.e., $m'_0 = d$ and $m'_1 = m_0$ (refer to Figure 3(a)). Otherwise, shift v_{m_1} to v_d(v_{m_0} unchanged) (refer to Figure 3(b)). Let $q = h(v_{m'_1})$. Since $\{v_{m_0}, v_d\} = \{v_{m'_0}, v_{m'_1}\}$ is a separating set of G, $t(v_{m'_0})$ and $t(v_{m'_1})$ are the last two tails as required by (C1).

At step (3.1), v_d is selected to be the last arc, for otherwise $K_{m'_0,m'_1}$ is not empty. For the example of Figure 3, if $v_d = v_{k_1}$, then $v_{k_2} \in K_{m'_0,m'_1}(= K_{m_0,k_1})$. Also notice that if $a(v_d)$ does not contain $h(v_{m_0})$ (refer to Figure 3(b)), then $K_{m'_0,m'_1}(= K_{d,m_0})$ is empty and the execution will proceed with step (4) after one more iteration. At step (7.2), we augment $T^{(i-1)}$ with an edge $v_{p(i)}\tilde{v}_{p(i),H_{p(i)}}$. Since $v_{p(i)}v_{p(1)}$ is not selected by our construction method as $|\tilde{N}_{G_{p(i)}}(v_{p(i)})| > 2$ and $h(v_{p(i)}) \in (h(v_{p(1)}), h(v_{p(2)}))$, we exclude $v_{p(1)}$ from $H_{p(i)}$ for this case.

Lemma 9 *$T^{(i)}$ obtained at step (7.2) contains an edge that connects $\tilde{v}_{p(i),H_{p(i)}}$ with another vertex in $\tilde{N}_{H_{p(i)}}(v_{p(i)})$.*

Proof. We first show $|\tilde{N}_{G_{p(i)}}(v_{p(i)})| \geq 2$ for $i \geq 3$ as follows. Notice that $d(v_{p(i)}) \geq 3$. If $|\tilde{N}_{G_{p(i)}}(v_{p(i)})| < 2$, then there exists $a(v_{p(t)})$ with $t > i$ that contains $h(v_{p(i)})$. Besides, both $a(v_{p(1)})$ and $a(v_{p(2)})$ do not contain $h(v_{p(i)})$. Without loss

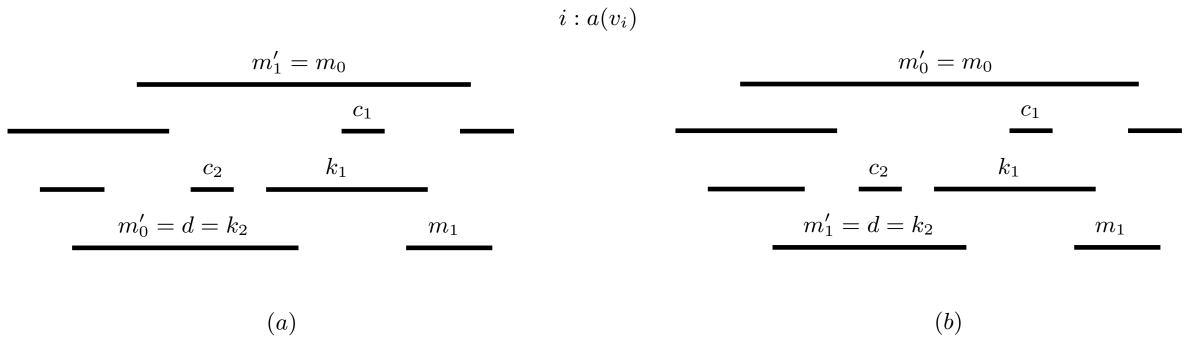

Figure 3: A feasible pair of $v_{m'_0}$ and $v_{m'_1}$. (a) When $a(v_d)$ contains $h(v_{m_0})$. (b) When $a(v_d)$ does not contain $h(v_{m_0})$.

of generality, suppose that $a(v_{p(2)})$ does not contain $h(v_{p(i)})$. Then, $a(v_{p(t)})$ contains $a(v_{p(2)})$, which is a contradiction to (C1).

Notice that $|\tilde{N}_{G_{p(i)}}(v_{p(i)})| \geq 3$ $(|\tilde{N}_{H_{p(i)}}(v_{p(i)})| \geq 2)$ or $h(v_{p(i)}) \notin (h(v_{p(1)}), h(v_{p(2)}))$ at step (7.2). When $h(v_{p(i)}) \notin (h(v_{p(1)}), h(v_{p(2)}))$, we can see that $|\tilde{N}_{H_{p(i)}}(v_{p(i)})| = |\tilde{N}_{G_{p(i)}}(v_{p(i)})| \geq 2$. Assume $v_{p(s)} = \tilde{v}_{p(i),H_{p(i)}}$, and let $v_{p(t)} \in \tilde{N}_{H_{p(i)}}(v_{p(i)}) - \{v_{p(s)}\}$. We first consider the situation of $s < t(< i)$. If $h(v_{p(i)}) \notin (h(v_{p(1)}), h(v_{p(2)}))$, then $h(v_{p(t)}) \notin (h(v_{p(1)}), h(v_{p(2)}))$. So, we have $H_{p(i)} = G_{p(i)}$ and $H_{p(t)} = G_{p(t)}$, which further implies $v_{p(s)} = \tilde{v}_{p(t),H_{p(t)}}$. If $h(v_{p(i)}) \in (h(v_{p(1)}), h(v_{p(2)}))$, then $v_{p(s)} = \tilde{v}_{p(t),H_{p(t)}}$(because $v_{p(s)} \neq v_{p(1)}$). We have $v_{p(t)}\tilde{v}_{p(t),H_{p(t)}} = v_{p(t)}v_{p(s)}$, which is contained in $T^{(t)} \subset T^{(i)}$.

Then we consider the situation of $t < s(< i)$. If $h(v_{p(i)}) \notin (h(v_{p(1)}), h(v_{p(2)}))$, then we have $H_{p(i)} = G_{p(i)}$ and $H_{p(s)} = G_{p(s)}$ similarly. The latter can assure that $\tilde{v}_{p(s),H_{p(s)}} = v_{p(t)}$ or $a(\tilde{v}_{p(s),H_{p(s)}})$ contains $t(v_{p(t)})$, which further implies $\tilde{v}_{p(s),H_{p(s)}} \in \tilde{N}_{G_{p(i)}}(v_{p(i)}) = \tilde{N}_{H_{p(i)}}(v_{p(i)})$. If $h(v_{p(i)}) \in (h(v_{p(1)}), h(v_{p(2)}))$, then $v_{p(s)} \neq v_{p(1)}$ and $v_{p(t)} \neq v_{p(1)}$, which implies $\tilde{v}_{p(s),H_{p(s)}} \in \tilde{N}_{H_{p(i)}}(v_{p(i)})$. We have $v_{p(s)}\tilde{v}_{p(s),H_{p(s)}}$ contained in $T^{(s)} \subset T^{(i)}$. □

Lemma 10 *There is an LCST of G if and only if Algorithm LCST-Circular-Arc-0 outputs $T^{(n)}$. Moreover, $T^{(n)}$ is such an LCST, which can be obtained in $O(n)$ time.*

Proof. We first assume that the algorithm terminates without producing $T^{(n)}$, i.e., $(m_0, m_1) = (s_0, s_1)$ holding at step (4), and there are r iterations executed for step (3). By $^{(i)}_{m_0}$ and $^{(i)}_{m_1}$ we denote the m_0 and m_1 used in the ith iteration, where $1 \leq i \leq r$. The m_0 and m_1 generated at step (3.2) in the ith iteration will serve as $^{(i+1)}_{m_0}$ and $^{(i+1)}_{m_1}$ in the $(i+1)$th iteration. We also use $K^{(i)}_{m_0,m_1}$ and $v^{(i)}_d$ to denote the K_{m_0,m_1} and v_d in the ith iteration. Now that $K^{(i+1)}_{m_0,m_1}$ is not empty, $a(v^{(i)}_d)$ contains $h(v^{(i)}_{m_0})$, i.e., $v^{(i+1)}_{m_1} = v^{(i)}_{m_0}$ ($K^{(i+1)}_{m_0,m_1}$ is empty if $a(v^{(i)}_d)$ does not contain $h(v^{(i)}_{m_0})$).

Construct a graph D with $V(D) = \{v^{(i)}_{m_0}, v^{(i)}_{m_1} | 1 \leq i \leq r\}$ and $E(D) = \{(v^{(i)}_{m_0}, v^{(i)}_{m_1}) | 1 \leq i \leq r\}$. Since $v^{(i+1)}_{m_1} = v^{(i)}_{m_0}$ for all $1 \leq i \leq r$ and $v^{(r)}_{m_0} = v^{(r+1)}_{m_1} = v_{s_1} = v^{(1)}_{m_1}$, D forms a cycle $(v^{(1)}_{m_1}, v^{(2)}_{m_1}, \ldots, v^{(r)}_{m_1}, v^{(1)}_{m_1})$ of length r. Notice that $(v^{(i+1)}_{m_0}, v^{(i+1)}_{m_1}) = (v^{(i)}_d, v^{(i)}_{m_0})$ is a separating set of G for all $1 \leq i \leq r$, where $(v^{(r+1)}_{m_0}, v^{(r+1)}_{m_1}) = (v_{s_0}, v_{s_1}) = (v^{(1)}_{m_0}, v^{(1)}_{m_1})$. It is implied by Lemma 1 that every edge of the cycle is an edge of any LCST of G, a contradiction.

Next we assume that the algorithm outputs $T^{(n)}$. We first show by induction that $T^{(i)}$ obtained at step (7.2) is an LCST of $G_{p(i)}$, where $3 \leq i \leq n$. Initially, $T^{(2)}$ obtained at step (6) is an LCST of $G_{p(2)}$. Suppose that $T^{(i-1)}$ is an LCST of $G_{p(i-1)}$. In order to show that $T^{(i)}$ is an LCST of $G_{p(i)}$, it suffices to show that both $G_{p(i)}[N_{T^{(i)}}(v_{p(i)})]$ and $G_{p(i)}[N_{T^{(i)}}(\tilde{v}_{p(i),H_{p(i)}})]$ are connected. Since $v_{p(i)}$ is a leaf vertex in $T^{(i)}$, $G_{p(i)}[N_{T^{(i)}}(v_{p(i)})]$ is connected. According to Lemma 9, $T^{(i)}$ contains an edge that connects $\tilde{v}_{p(i),H_{p(i)}}$ with a neighbor of $v_{p(i)}$ in $H_{p(i)}$. Since $T^{(i-1)}$ is an LCST of $G_{p(i-1)}$, $G_{p(i-1)}[N_{T^{(i-1)}}(\tilde{v}_{p(i),H_{p(i)}})]$ is connected. It follows that $G_{p(i)}[N_{T^{(i)}}(\tilde{v}_{p(i),H_{p(i)}})]$ is connected.

We then show that $T^{(n)}$ obtained at step (7.1) is also an LCST of $G_{p(n)}$. Now that $v_{p(1)}v_{p(2)} \in E(T^{(n)})$, it suffices to show $G[\{v_{p(2)}, v_{p(i)}, v_{p(i+1)}, \ldots, v_{p(n)}\}]$ is connected. Since $|\tilde{N}_{G_{p(i)}}(v_{p(i)})| = 2$ and $h(v_{p(i)}) \in (h(v_{p(1)}), h(v_{p(2)}))$, we have $v_{p(1)} \in \tilde{N}_{G_{p(i)}}(v_{p(i)})$. Besides, $h(v_{p(i+1)}), h(v_{p(i+2)}), \ldots, h(v_{p(n)})$ all belong to $(h(v_{p(1)}), h(v_{p(2)}))$. Assume that $v_{p(f)}$ is the other vertex in $\tilde{N}_{G_{p(i)}}(v_{p(i)})$. If $v_{p(f)} = v_{p(2)}$, $G[\{v_{p(2)}, v_{p(i)}, v_{p(i+1)}, \ldots, v_{p(n)}\}]$ is connected, as a consequence of $d(v) \geq 3$ for every $v \in V(G)$. If $v_{p(f)} \neq v_{p(2)}$, then $G[\{v_{p(2)}, v_{p(i)}, v_{p(i+1)}, \ldots, v_{p(n)}\}]$ is also connected, for otherwise there is a contradiction to (C2).

Finally, we discuss the time complexity of the algorithm. With F, steps (1) and (5) can be completed in $O(n)$ time. Notice that $v_k \in K_{m_0,m_1}$ if and only if there exist two vertices v_a and v_b with $d(v_a) = d(v_b) = 3$ satisfying that $h(v_a) \in (h(v_{m_0}),\ h(v_{m_1}))$, $h(v_b) \in (h(v_{m_0}), h(v_{m_1}))$ and $h(v_a), h(v_b) \in a(v_{m_0}) \cap a(v_k)$. The worst case of step (3) happens as the circle is traversed clockwise, starting from the pair of $a(v_{s_0})$ and $a(v_{s_1})$, and then returning to the original pair. Throughout the execution of step (3), $O(n)$ arcs are examined in order to find K_{m_0,m_1}, $v_{p(1)}$ and $v_{p(2)}$. Since $\tilde{v}_{p(i),H_{p(i)}} = \tilde{v}_{p(i-1),H_{p(i-1)}}$ or $\tilde{v}_{p(i),H_{p(i)}} = v_{p(i-1)}$ for $i \geq 4$, $\tilde{v}_{p(i),H_{p(i)}}$ can be easily determined in $O(1)$ time in each iteration of step (7). Hence, it takes $O(n)$ time to complete step (7). The other steps can be completed in $O(1)$ time. $\square$

4 Conclusion

In this paper, we have presented an optimal algorithm that can determine whether a circular-arc graph G contains an LCST or not, and construct it, if it exists. Given an intersection model of G, the algorithm requires $O(n)$ time and $O(n)$ space. It was shown in [3, 17] that an interval graph has an LCST if and only if it is 2-connected. In order to construct an LCST of G, it is natural to divide G into 2-connected interval subgraphs such that their LCSTs can collectively form an LCST of G. Since G having $d(v) = 1$ for some vertex v is an interval graph, we only need to consider G with $d(v) \geq 2$ for all vertices v. Further, according to Lemma 2, only G that has $d(v) = 2$ for at most three vertices v has to be considered.

It is known that a 2-connected interval graph has $d(v) = 2$ for exactly one vertex v. In other words, if an interval graph has $d(v) = 2$ for two or more vertices v, then it has no LCST. So, each 2-connected interval subgraph of G should have $d(v) = 2$ for exactly one vertex v, and dividing G into 2-connected interval subgraphs heavily relies on the number of vertices v in G that have $d(v) = 2$. As a consequence, we considered four situations when G has $d(v) = 2$ for $0, 3, 2$ and 1 vertex v, respectively.

References

[1] K. S. Booth and G. S. Lueker. Testing for the consecutive ones property, interval graphs, and graph planarity using *PQ*-tree algorithms. *J. Comput. System Sci.*, 13(3):335–379, 1976.

[2] L. Cai. On spanning 2-trees in a graph. *Discrete Appl. Math.*, 74(3):203–216, 1997.

[3] L. Cai. The complexity of the locally connected spanning tree problem. *Discrete Appl. Math.*, 131(1):63–75, 2003.

[4] G. Chang and G. Nemhauser. The *k*-domination and *k*-stability problems on sun-free chordal graphs. *SIAM J. Algebraic Discrete Methods*, 5:332–345, 1984.

[5] M.-S. Chang. Efficient algorithms for the domination problems on interval and circular-arc graphs. *SIAM J. Comput.*, 27(6):1671–1694, 1998.

[6] P. F. Dietz. *Intersection graph algorithms.* PhD thesis, Comp. Sci. Dept., Cornell University, Ithaca, NY, 1984.

[7] M. Farber. Characterizations of strongly chordal graphs. *Discrete Math.*, 43(2-3):173–189, 1983.

[8] A. M. Farley. Networks immune to isolated failures. *Networks*, 11(3):255–268, 1981.

[9] A. M. Farley and A. Proskurowski. Networks immune to isolated line failures. *Networks*, 12(4):393–403, 1982.

[10] M. C. Golumbic. *Algorithmic graph theory and perfect graphs*, volume 57 of *Annals of Discrete Mathematics*. Elsevier Science B.V., Amsterdam, second edition, 2004. With a foreword by Claude Berge.

[11] M. C. Golumbic and P. L. Hammer. Stability in circular arc graphs. *J. Algorithms*, 9(3):314–320, 1988.

[12] U. I. Gupta, D. T. Lee, and J. Y.-T. Leung. Efficient algorithms for interval graphs and circular-arc graphs. *Networks*, 12(4):459–467, 1982.

[13] W. L. Hsu and J. P. Spinrad. Independent sets in circular-arc graphs. *J. Algorithms*, 19(2):145–160, 1995.

[14] W. L. Hsu and K.-H. Tsai. Linear time algorithms on circular-arc graphs. *Inform. Process. Lett.*, 40(3):123–129, 1991.

[15] J. M. Keil and D. Schaefer. An optimal algorithm for finding dominating cycles in circular-arc graphs. *Discrete Appl. Math.*, 36(1):25–34, 1992.

[16] D. T. Lee, M. Sarrafzadeh, and Y. F. Wu. Minimum cuts for circular-arc graphs. *SIAM J. Comput.*, 19(6):1041–1050, 1990.

[17] C.-C. Lin, G. J. Chang, and G.-H. Chen. Locally connected spanning trees in strongly chordal graphs and proper circular-arc graphs. *Discrete Math.* accepted.

[18] S. Masuda and K. Nakajima. An optimal algorithm for finding a maximum independent set of a circular-arc graph. *SIAM J. Comput.*, 17(1):41–52, 1988.

[19] R. M. McConnell. Linear-time recognition of circular-arc graphs. *Algorithmica*, 37(2):93–147, 2003.

[20] R. Paige and R. E. Tarjan. Tree partition refinement algorithms. *SIAM J. Comput.*, 16:973–989, 1987.

[21] K. H. Tsai and D. T. Lee. k best cuts for circular-arc graphs. *Algorithmica*, 18(2):198–216, 1997.

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