An Optimal Algorithm for Finding Articulation Vertex of Circular Permutation Graphs

Hirotoshi HONMA[1] Yoko NAKAJIMA[1]

Abstract: Let $G_s = (V_s, E_s)$ be a simple connected graph. A vertex $v \in V_s$ is an articulation vertex if deletion of v and its incident edges from G_s disconnects the graph into at least two connected components. Finding all articulation vertices of a given graph is called the articulation vertex problem. This problems can be applied to improve the stability and robustness of communication network systems. In this paper, we propose a linear time algorithm for the articulation vertex problem of circular permutation graphs.

Key words: Design and analysis of algorithms, Circular Permutation Graphs, Articulation Vertices;

1 Introduction

Let $G_s = (V_s, E_s)$ be a simple connected graph with $|V| = n$ and $|E| = m$. A vertex $v \in V_s$ is an *articulation vertex* if the deletion of v and its incident edges from G_s disconnects the graph into at least two connected components. A graph with no articulation vertex is called a *biconnected graph*. Finding all articulation vertices of a given graph is called the articulation vertex problem. An $O(n+m)$ time algorithm exists for solving the articulation vertex problem in simple graphs by using the traditional depth-first spanning tree method [1]. Moreover, efficient parallel algorithms for finding articulation vertices, bridges, and biconnected components in general graphs are given in [2, 3]. This problem can be applied to improve the stability and robustness of communication network systems [4].

In many cases, more efficient algorithms can be developed by restricting the classes of graphs. For instance, for *permutation graphs*, Ibarra and Zheng [5] proposed an $O(\log n)$ time parallel algorithm using $O(n/\log n)$ processors for the articulation vertex problem. Furthermore, for *interval graphs*, Sprague and Kulkarni [6] proposed an $O(\log n)$ time parallel algorithms with $O(n/\log n)$ processors for the articulation vertex problem. Kao and Horng [7] proposed optimal $O(\log n)$ time parallel algorithms with $O(n/\log n)$ processors for finding all articulation vertices, bridges, and biconnected components of *circular-arc graphs*, which are a superclass of interval graphs.

Let $V_p = [1, 2, \ldots, n]$ be a vertex set and $P = [p(1), p(2), \ldots, p(n)]$ be a permutation of V_p. A permutation graph G_p is visualized by its corresponding *permutation model* M_p, which consists of two horizontal parallel lines called the *top channel* and *bottom channel*, respectively. Place the vertices $1, 2, \ldots, n$ on the top channel, ordered from left to right, and similarly, place $p(1), p(2), \ldots, p(n)$ on the bottom channel. Next, for each $i \in V_p$, draw a straight line from i on the top channel to i on the bottom channel. Then, an edge (i, j) in G_p exists if and only if lines i and j intersect in M_p. In this paper, "line" and "vertex" are used interchangeably. An example of a permutation model M_p and its corresponding permutation graph G_p is shown in Fig. 1. Permutation graphs are an important subclass of perfect graphs, and they are used for modeling practical problems in many areas, such as biology, genetics, very large scale integration (VLSI) design, and network planning [8].

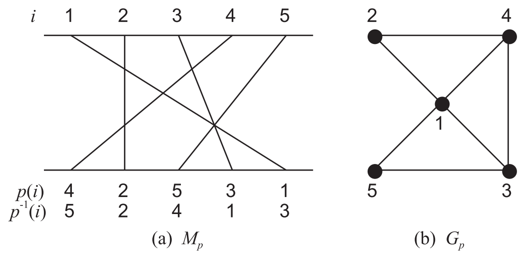

Figure 1: Permutation model M_p and graph G_p.

Circular permutation graphs properly contain a set of permutation graphs as a subclass. Rotem and Urrutia first introduced circular permutation graphs and provided an $O(n^{2.376})$ time recognition algorithm [9]. Lou and Sarrafzadeh showed that circular permutation graphs and their models have several applications in VLSI layout design [10]. They presented an $O(\min(\delta n \log\log n, n \log n) + |E|)$ time algorithm for finding a maximum independent set of a circular permutation model, where δ is the minimum degree of vertices in the corresponding circular permutation graph. Furthermore, they presented an $O(n \log\log n)$ time algorithm for finding the maxi-

[1]Department of Information Engineering, Kushiro National College of Technology

mum clique and the chromatic number of a circular permutation model. Subsequently, the recognition algorithm was improved in $O(m+n)$ time by Sritharan [11].

In this paper, we propose linear time algorithms for the articulation problems in circular permutation graphs. The rest of this paper is organized as follows. Section 2 describes some definitions of circular permutation graphs and models. Section 3 introduces the extended circular permutation model and its properties. Sections 4 consider algorithms that address articulation vertex problem and the complexity of this algorithm. Section 5 concludes this paper.

2 Circular Permutation Model and Graph

We first illustrate the *circular permutation model* before defining the circular permutation graph. There exist inner and outer circles C_1 and C_2 with radii $r_1 < r_2$. Let $CP = [cp(1), cp(2), \ldots, cp(n)]$ be a permutation of integer sequence $[1, 2, \ldots, n]$. Furthermore, $cp^{-1}(i)$, $1 \leqslant i \leqslant n$, denotes the position of the number i in CP. Consecutive integers i, $1 \leqslant i \leqslant n$, are set to be counter-clockwise on C_1. Similarly, $cp(i)$, $1 \leqslant i \leqslant n$, is set to be counter-clockwise on C_2. For each i, $1 \leqslant i \leqslant n$, draw a chord joining the two i's, one on C_1 and the other on C_2, denoted as *chord i*. The geometric representation described above is called a circular permutation model *CM*. Figure 2 illustrates an example of *CM* with 12 chords constructed by $CP = [11, 1, 5, 10, 2, 7, 6, 9, 4, 8, 3, 12]$. This model is considered to be *proper* if any two chords i and j intersect at most once in the *CM*. In this paper, we consider only proper circular permutation graphs and models, and therefore, the word "proper" is omitted henceforth.

Next, we introduce circular permutation graphs. An undirected graph G is a circular permutation graph if it can be represented by the following circular permutation model *CM*: each vertex of the graph corresponds to a chord in the annular region between two concentric circles C_1 and C_2, and two vertices are adjacent in G if and only if their corresponding chords intersect exactly once [9]. Figure 3 illustrates the circular permutation graph G corresponding to *CM* shown in Fig. 2. In this example, $\{2, 10\}$ is an articulation vertex set.

Next, we consider a fictitious chord $\overline{a}$ which connects the point a' that is placed between 1 and 12 on C_1 and point a'' on C_2. A chord that intersects $\overline{a}$ is called a *feedback chord*. The set of all feedback chords is denoted by F. Moreover, a set of feedback chords that intersect $\overline{a}$ in clockwise is defined as F^-, and a set of feedback chords that intersect $\overline{a}$ counterclockwise is defined as F^+. We must place point a'' on

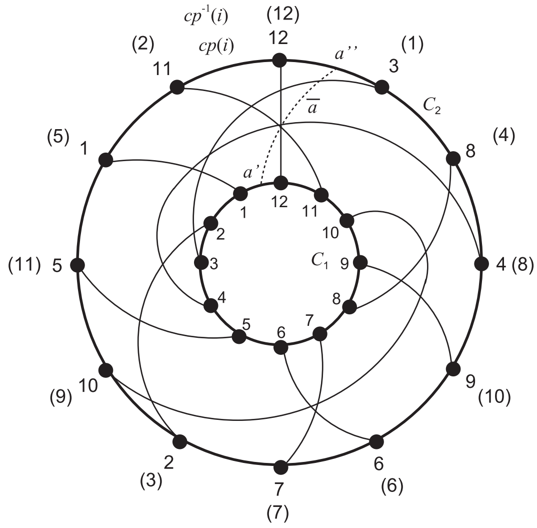

Figure 2: Circular permutation model *CM*.

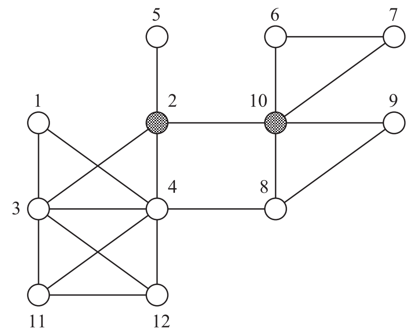

Figure 3: Circular permutation graph G.

C_2 so that $|F^-| = |F^+|$ is satisfied. In the example shown in Fig. 2, point a'' is placed between 3 and 12 on C_2. Consequently, $F = \{3, 4, 11, 12\}$, $F^- = \{3, 4\}$ and $F^+ = \{11, 12\}$. If a fictitious chord $\overline{a}$ exists that does not intersect any chord in *CM*, a model formed by opening *CM* along $\overline{a}$ is equivalent to a permutation model. This problem can be solved by applying Ibarra et al.'s algorithm [5] because this problem is the same as that of permutation graphs. In this paper, we assume that any fictitious chord intersects at least one chord.

3 Extended Circular Permutation Model

In this section, we introduce an *extended circular permutation model ECM* that is constructed from a *CM*.

Let n be the number of chords in *CM*. First, a point a' is fixed between 1 and n on C_1. Next, we consider a fictitious chord $\overline{a}$ with $|F^-| = |F^+|$. In Fig. 2, we obtain $|F^-| = |F^+| = 2$ by placing point a'' between 3 and 12 on C_2. *ECM* is formed by opening *CM* along

$\overline{a}$. *ECM* consists of two horizontal parallel lines L_1 and L_2, called top and bottom channels, respectively. The top channel L_1 is assigned the consecutive number i, $-n+1 \leqslant i \leqslant 2n$, from left to right. The bottom channel L_2 is assigned $p(i)$, $-n+1 \leqslant i \leqslant 2n$, from left to right. Here, $p(i)$, $1 \leqslant i \leqslant n$, on L_2, is assigned a *cp* value on C_2 in the counter-clockwise direction from point a''. Next, $p(i)$, $1 \leqslant i \leqslant n$, changes to $p(i) - n$ if $i \in F^+$. Furthermore, $p(i)$, $1 \leqslant i \leqslant n$, changes to $p(i) + n$ if $i \in F^-$. We execute $p(i-n) = p(i) - n$ and $p(n+i) = p(i) + n$ for $1 \leqslant i \leqslant n$. For each $-n+1 \leqslant i \leqslant 2n$, a straight line is drawn from i on L_1 to i on L_2. After executing the above process, *ECM* is constructed from *CM*. Figure 4 illustrates *ECM* constructed from *CM* shown in Fig. 2. Here, $p^{-1}(i)$ denotes the position of i on L_2.

Circular permutation and circular-arc graphs are circular versions of permutation and interval graphs, respectively. Moreover, as mentioned in Section 1, circular permutation and circular-arc graphs are superclasses of permutation and interval graphs, respectively. Efficient algorithms have been developed that address various problems concerning permutation and circular-arc graphs. However, in general, problems for circular graphs tend to be more difficult than those for non-circular graphs. One of the reasons is that we can not uniquely determine the starting position of an algorithm for a circular graph due to the existence of feedback elements although it can be fixed for a non-circular graphs.

For several problems, we can develop circular versions of the existing algorithms by constructing extended intersection models of the problems. By using extended intersection models, we can determine a start position of algorithm uniquely and apply partially the algorithms of the non-circular versions. For instance, this method has been applied to develop efficient algorithms for the shortest path query problem [12, **?**], the articulation vertex problem [7] on circular-arc graphs, maximum clique and chromatic number problems [10], the spanning forest problem [13] on circular-permutation graphs. In this paper, we use *ECM* to construct an efficient algorithm for an articulation vertex problem.

Property 1 stated below, can be derived in a straightforward manner from the processes of constructing *ECM*.

Property 1 *Lines $i-n$, i, and $i+n$ in ECM correspond to the vertex i in G.*

Two vertices i and j are adjacent in a circular permutation graph if and only if their corresponding chords intersect exactly once in *CM*. When two chords i and j $(i < j)$ intersect in *CM*, we distinguish the following three cases:

Case 1: $i \in F^-$ or $j \in F^+$

In this case, lines j and $i+n$ intersect in *ECM* with lines $i+n$ and j, respectively.

Case 2: $i \in F^+$ and $j \in F^-$

This case is infeasible because it implies that chords i and j intersect twice in *CM*.

Case 3: Remaining conditions for i and j

In these cases, lines i and j intersect in *ECM*.

Based on the above mentioned information, we can state Property 2 as follows:

Property 2 *Let i and j $(i < j)$ be two vertices in G. Then, vertex i is adjacent to j if and only if lines i and j, or lines i and $j-n$, or lines $i+n$ and j intersect in ECM.*

Some notations that form the basis of the algorithms in sections 4 and 5 are defined as follows: The set of all lines that intersect line i in *ECM* is denoted by $N(i)$. In addition, $N[i] = N(i) \cup \{i\}$. For line i in *ECM*, the following functions are defined: $TR(i) = \max\{\ j \mid j \in N[i]\}$ and $STR(i) = \max\{\ j \mid j \in (N[i] \setminus TR(i)) \cup \{i\}\}$. $BR(i) = k$ such that $p^{-1}(k) = \max\{\ p^{-1}(j) \mid j \in N[i]\}$. $A(i)$ and $B(i)$ for line i are defined as follows: $A(i) = |\{\ j \mid j \leqslant i, p^{-1}(j) > i\}|$ and $B(i) = |\{\ j \mid j > i, p^{-1}(j) \leqslant i\}|$. Table 1 shows $TR(i)$, $BR(i)$, $A(i)$ and $B(i)$ for *ECM* shown in Fig. 4.

4 Algorithm AVC

In this section, we present an algorithm AVC that finds all articulation vertices of a circular permutation graph. Let *ECM* be an extended circular permutation model constructed from *CM*. We say a *path* exists between i and j if either line i directly intersects line j, or there exist lines $k_1, k_2, \ldots, k_s$ in *ECM* such that line i intersects k_1, k_1 intersects k_2, $\ldots$, k_{s-1} intersects k_s, and k_s intersects line j. Moreover, two lines i and j in *ECM* belong to the same *line component* if there exists a path between i and j. In Fig. 4, line 8 is a cut line for lines 10 and 11.

4.1 Properties of Articulation Vertex

Ibarra and Q. Zheng [5] provided Lemma 1, which is a necessary and sufficient condition for the articulation vertex in a permutation graph G_p.

Lemma 1 ([5]) *Let G_p be a permutation graph corresponding to a permutation model M_p. A vertex v is an articulation vertex of G_p if and only if there exists an integer i $(1 \leqslant i \leqslant n)$ such that either of the following conditions holds in M_p:*

(1) $v = TR(p(i))$ for $B(i) = 1$, $A(i-1) = 1$, and $p(i) < i$,

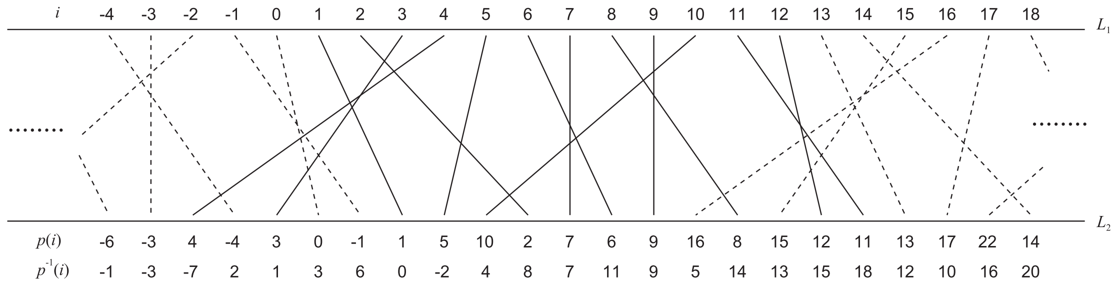

Figure 4: Extended circular permutation model *ECM*.

Table 1: Example of $TR(i)$, $BR(i)$, $A(i)$ and $B(i)$.

i	-3	-2	-1	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
$p(i)$	-3	4	-4	3	0	-1	1	5	10	2	7	6	9	16	8	15	12	11	13	17
$p^{-1}(i)$	-3	-7	2	1	3	6	0	-2	4	8	7	11	9	5	14	13	15	18	12	10
$TR(i)$	-2	-2	4	4	4	10	4	4	5	10	10	16	10	10	16	16	16	22	16	16
$BR(i)$	-4	-4	-1	-1	1	2	2	2	2	6	6	8	8	8	11	11	13	14	12	11
$A(i)$				2	2	2	1	1	1	1	1	1	1	1	1	2				
$B(i)$				2	2	2	1	1	1	1	1	1	1	1	1	2				
$av_1(i)$										10		10			16					
$av_2(i)$								2	2					8						

(2) $v = BR(i)$ *for* $A(i) = 1$, $B(i-1) = 1$, *and* $p^{-1}(i) < i$.

Let $G = (V, E)$, $|V| = n$ be a circular permutation graph corresponding to a circular permutation model *CM*, and *ECM* be an extended circular permutation model constructed from *CM*. Hence, Lemmas 2 and 3 follow from Lemma 1.

Lemma 2 $TR(p(i))$ *is a cut line for* $i-1$ *and* i *in ECM if* $B(i) = 1$, $A(i-1) = 1$ *and* $p(i) < i$.

(Proof) By Lemma 1–(1), the elimination of line $TR(p(i))$ from *ECM* disconnects it into at least two line components when $B(i) = 1$, $A(i-1) = 1$ and $p(i) < i$. Assume that *ECM* is divided into two line components, namely M_1 and M_2 by removing line $TR(p(i))$ (Fig. 5). We show that M_1 and M_2 include lines $i-1$ and line i, respectively.

From condition $A(i-1) = 1$, *ECM* has a line $j(\leqslant i-1)$ with $p^{-1}(j) > i-1$. We assume that $p^{-1}(j) > i$. There exists some line $r(\leqslant i-1)$ with $p^{-1}(r) = i$ from condition $p(i) < i$. It follows $A(i-1) \geqslant 2$ and contradicts the hypothesis of $A(i-1) = 1$. Hence, such a line r does not exist. This implies that $p^{-1}(j) = i$ and line j has maximum p^{-1} value in M_1.

According to Lemma 1–(1), only line $TR(p(i))$ connects M_1 and M_2. Furthermore, by the condition $B(i) = 1$, $TR(p(i)) > i$ and $p^{-1}(TR(p(i))) < i$. Since *ECM* is constructed under the condition that $|F^-| = |F^+|$, there are i positions from 1 to i on L_2. However, $i+1$ positions are required from 1 to i on L_2 when $p^{-1}(i) < i$. This is the contradiction of the pigeonhole principle. Thus, $p^{-1}(i) > i$.

Hence, for two lines $i-1$ and i, $p^{-1}(i-1) < i$ and $p^{-1}(i) > i$, respectively. Furthermore, line $p(i)$ has maximum p^{-1} value in M_1 and only line $TR(p(i))$ connects M_1 and M_2. Hence, M_1 and M_2 include lines $i-1$ and i, respectively. □

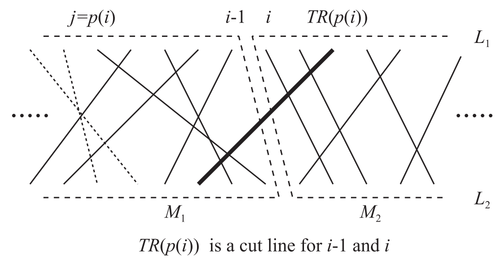

Figure 5: Example of Lemma 2.

Lemma 3 $BR(i)$ *is a cut line for* i *and* $i+1$ *in ECM if* $A(i) = 1$, $B(i-1) = 1$, *and* $p^{-1}(i) < i$.

(Proof) Lemma 3 is symmetric to Lemma 2. Hence, its proof is similar to that of Lemma 2. □

Lemma 4 *Let* $G = (V, E)$ *be a circular permutation graph corresponding to ECM. A vertex* v *is an articulation vertex of* G *if and only if elimination of line* v *disconnects ECM into at least three line components.*

(Proof) Sufficiency of this condition obviously holds; thus, we only prove necessity. Consider a case of

Algorithm 1: Algorithm AVC

Input: $CP = \{p(1), p(2), \ldots, p(n)\}$ of a circular permutation graph G.
Output: Articulation vertices of G.
(Step 1)
Construct ECM and compute $p^{-1}(i)$;
(Step 2)
Compute $TR(i)$, $BR(i)$ for i in ECM;
(Step 3)
Compute $A(i)$ and $B(i)$ for i in ECM;
(Step 4)
/* Compute $av_1(i)$ */ ;
for *each* $1 \leqslant i \leqslant n$ **do**
  if $(B(i) = 1$ *and* $A(i-1) = 1$ *and* $p(i) < i)$ **then** $av_1(i) = TR(p(i))$;
end
(Step 5)
/* Compute $av_2(i)$ */;
for *each* $1 \leqslant i \leqslant n$ **do**
  if $(A(i) = 1$ *and* $B(i-1) = 1$ *and* $p^{-1}(i) < i)$ **then** $av_2(i) = BR(i)$;
end
(Step 6)
for *each* $1 \leqslant i \leqslant n$ **do**
  Normalize $av_1(i)$;
  Normalize $av_2(i)$;
end
(Step 7)
if $av_1(i)$ *has at least two same values for* $1 \leqslant i \leqslant n$ **then** $av_1(i)$ is an articulation vertex ;
if $av_2(i)$ *has at least two same values for* $1 \leqslant i \leqslant n$ **then** $av_2(i)$ is an articulation vertex ;

Function Normalize v\{
if $v < 1$ **then** $v := v + n$;
if $v > n$ **then** $v := v - n$;
return v ;
\}

where ECM is divided into just two line components M_1 and M_2 by removing line v from ECM. M_1 includes some copies of lines that are in M_2, and M_2 includes some copies of lines that are on M_1 subject to conditions $F \neq \emptyset$ and $|F^-| = |F^+|$. Thus, ECM is divided into M_1 and M_2, but a graph corresponding to $M_1 \cup M_2$ is connected.

In the following lemma, assume that ECM is divided into $k(\geqslant 3)$ line components $M_1, M_2, \ldots, M_k$ when line v is removed from ECM. Here, M_1 includes some copies of lines that are in M_k, and M_k also includes some copies of lines that are in M_1. Thus, the subgraph corresponding to $M_1 \cup M_k$ is connected. Hence, $G - v$ is a graph with $k-1$ connected components $(M_2, \ldots, M_{k-1}, M_1 \cup M_k)$. That is, $G - v$ is disconnected. □

Lemma 5 *Let $G = (V, E)$ be a circular permutation graph corresponding to ECM. A vertex v is an articulation vertex of G if and only if there exist at least two identical values of v for $1 \leqslant i \leqslant n$ such that either of the following two conditions holds in ECM;*

(1) $v = TR(p(i))$ for $B(i) = 1$ and $A(i-1) = 1$ and $p(i) < i$.

(2) $v = BR(i)$ for $A(i) = 1$ and $B(i-1) = 1$ and $p^{-1}(i) < i$.

(Proof) Assume that condition (1) holds for i_1 and i_2, i.e., $v = TR(p(i_1)) = TR(p(i_2))$ for $1 \leqslant i_1 < i_2 \leqslant n$. By Lemma 2, v is a cut line for $i_1 - 1$ and i_1, and is also a cut line for $i_2 - 1$ and i_2. Hence, the elimination of line v disconnects ECM into three line components M_1, M_2, and M_3 that include $i_1 - 1$, i_1, and i_2, respectively. By Lemma 4, G is disconnected because ECM is divided into at least three line components. Thus, vertex v is an articulation vertex of G. In a similar manner, we can prove case (2). □

We show an example in which vertex 10 is recognized as an articulation vertex by applying Lemma 5. In Fig. 4, when $i = 6$, $B(i) = 1$, $A(i-1) = 1$, and $p(i) = 2 < i$, and consequently, $v = TR(p(i)) = TR(p(6)) = 10$. Similarly, when $i = 8$, $B(i) = 1$, $A(i-1) = 1$, and $p(i) = 6 < i$, and thus, $v = TR(p(i)) = TR(6) = 10$ holds true. Thus, we can obtain 10 as the articulation vertex because the values ($v = 10$) appear for $i = 6$ and 8.

4.2 Analysis of Algorithm AVC

The algorithm used to find all articulation vertices of a circular permutation graph is described formally in Algorithm AVC.

Next, we analyze the complexity of Algorithm AVC. In Step 1, we construct a circular permutation model ECM that can be executed in $O(n)$ time. In Step 2, $TR(i)$ and $BR(i)$ are computed. In Step 3, $A(i)$ and $B(i)$ are obtained. The above preprocessing steps take $O(n)$ time [5]. Steps 4–6 compute $av_1(i)$ and $av_2(i)$ by applying Lemma 5 and they run in $O(n)$ time. By applying Step 6 of Algorithm AVC, we obtain $av_1(i)$ and $av_2(i)$ in Table 1. After executing Step 7, all articulation vertices of a circular permutation graph are correctly found. In Table 1, each of av_1 and av_2 has two identical values, 10 and 2, respectively. Thus, vertices 10 and 2 are articulation vertices. Hence, we obtain the following theorem:

Theorem 1 *Algorithm AVC can solve the articulation vertex problem of circular permutation graph in $O(n)$ time.*

5 Concluding Remarks

In this paper, we proposed an algorithm that runs in $O(n)$ time to find all articulation vertices of a circular permutation graph. Our algorithm is constructed by employing Ibarra's algorithm [5]. In future, we will continue this research by extending the results to other classes of graphs.

Acknowledgments

This research is supported by the Grant-in-Aid of Japan Society for the Promotion of Science for Scientific Research (C) 22500020.

References

[1] D. Jungnickel, Graphs, Networks and Algorithms, Springer, 2000.

[2] R.E. Tarjan and U. Vishkin, "An efficient parallel biconnectivity algorithm," SIAM J. Comput., vol.14, no.4, pp.862–874, 1985.

[3] Y.H. Tsin and F.Y. Chin, "Efficient parallel algorithm for a class of graph theoretic problems," SIAM J. Comput., vol.13, no.3, pp.580–599, 1984.

[4] H. Honma and S. Masuyama, "A parallel algorithm for finding all hinge vertices of an interval graph," IEICE Trans. Inf. & Syst., vol.E84-D, no.3, pp.419–423, 2001.

[5] O.H. Ibarra and Q. Zheng, "Finding articulation points and bridges of permutation graphs," International Conference on Parallel Processing, St. Charles, Illinois, pp.77–81, 1993.

[6] A.P. Sprague and K.H. Kulkarni, "Optimal parallel algorithms for finding cut vertices and bridges of interval graphs," Inf. Process. Lett., vol.42, no.2, pp.229–234, 1992.

[7] T.W. Kao and S.J. Horng, "Optimal algorithms for computing articulation points and some related problems on a circular-arc graph," Parallel Computing, vol.21, no.6, pp.953–969, 1995.

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

[9] D. Rotem and J. Urrutia, "Circular permutation graphs," Networks, vol.12, no.4, pp.429–437, 1982.

[10] R.D. Lou and M. Sarrafzadeh, "Circular permutation graph family with applications," Discrete Appl. Math., vol.40, no.3, pp.433–457, 1992.

[11] R. Sritharan, "An linear time algorithm to recognize circular permutation graphs," Networks, vol.27, no.3, pp.171–174, 1996.

[12] D. Chen, D.T. Lee, R. Sridhar, and C. Sekharam, "Solving the all-pair shortest path query on interval and circular-arc graphs," Networks, vol.31, pp.249–258, 1998.

[13] H. Honma, S. Honma, and S. Masuyama, "An optimal parallel algorithm for constructing a spanning tree on circular permutation graphs," IEICE Trans. Inf. & Syst., vol.E92-D, no.2, pp.141–148, 2009.