SRC TR 88-53

Circular-Arc Containment Graphs

by

M. V. Nirkhe, S. Masuda and K. Nakajima

Circular-Arc Containment Graphs *

Madhura V. Nirkhe

Electrical Engineering Department and Systems Research Center

University of Maryland, College Park, MD 20742

Sumio Masuda †

Department of Information and Computer Sciences,

Osaka University, Toyonaka, Osaka 560, Japan

and

Kazuo Nakajima

Electrical Engineering Department, Institute for Advanced Computer Studies

and Systems Research Center

University of Maryland, College Park, MD 20742

July 25, 1988.

*This research was supported in part by National Science Foundation grants MIP-84-51510 and CDR-85-00108 and in part by a grant from AT&T.

†The work of this author was done while he was visiting the Electrical Engineerig Department and Systems Research Center at the University of Maryland, College Park, MD 20742.

Circular-Arc Containment Graphs

by

Madhura V. Nirkhe, Sumio Masuda and Kazuo Nakajima

Abstract

We introduce a new class of containment graphs called circular-arc containment graphs. A graph is called a circular-arc containment graph if there exists a family of arcs on a circle, such that each vertex corresponds to an arc and two vertices are connected by an edge if and only if one of the corresponding arcs contains the other. We characterize this class of graphs by establishing its equivalence to another relatively new class of intersection graphs, called circular permutation graphs. Given a circular-arc containment graph in the form of a family of arcs on a circle, we describe efficient algorithms for finding a maximum clique and a maximum independent set of the graph.

1 Introduction

A graph $G = (V, E)$ is called an *intersection graph* for a family F of sets if there exists a one-to-one correspondence between V and F such that two vertices in V are adjacent if and only if the corresponding sets in F have a nonempty intersection. If F is a family of line segments in a permutation diagram, G is called a *permutation graph.* This class of graphs has widely been studied by a number of researchers [1,4,5,7,11,13]. As a generalization of a permutation graph, Rotem and Urrutia [12] introduced a *circular permutation graph* (abbreviated as *CPG*), which is the intersection graph for a family of chords in a circular permutation diagram. They developed a recognition algorithm for CPGs, which runs in $O(n^{2.49+})$ time by using a comparability graph recognition algorithm of the same complexity [13], where n is the number of vertices.

A graph $G = (V, E)$ is called a *containment graph* for a family F of sets if there exists a one-to-one correspondence between V and F such that two vertices in V are adjacent if and only if one of the corresponding sets in F is a subset of the other. We denote by G_F the containment graph for F. If F is a family of intervals on the real line, G_F is called an *interval containment graph* (abbreviated as *ICG*). It is known that every ICG is a permutation graph and vice versa [2,8]. The problems of finding a maximum clique and a maximum independent set are solvable in polynomial time [7] by using the algorithm for finding a longest increasing subsequence of a sequence [6]. Therefore they are also polynomially solvable for ICGs [9].

A *circular-arc containment graph* (abbreviated as *CACG*) is the containment graph for a family of arcs on a circle. In this paper we investigate this class of graphs. We first prove that every CACG is a CPG and vice versa. This result leads to an $O(n^{2.49+})$ time recognition algorithm for CACGs.

We then consider the problems of finding a maximum clique and a maximum in-

dependent set of a CACG assuming that the graph is given in the form of a family of arcs on a circle. Every CACG is a comparability graph. Given a comparability graph with a transitive orientation, one can find a maximum clique of the graph in $O(n+e)$ time [4,7], where n and e are the number of vertices and edges, respectively, in the graph. Although finding a transitive orientation takes $O(n^2)$ time for a general comparability graph [13], it can be done in $O(nlogn)$ time for a CACG if the corresponding family of arcs is given [10]. Thus, it is straightforward to construct an $O(nlogn + e)$ time algorithm for finding a maximum clique of a CACG. In this paper, we present a simpler algorithm that runs in $O(nlogn)$ time.

Given an n vertex comparability graph with a transitive orientation, the maximum independent set problem can be transformed to a maximum 0-1 flow problem [7], and thus is solvable in $O(n^{2/3}e)$ time [3]. We show that, for a CACG, we can find a maximum independent set in $O(n\delta logn + e)$ time, where δ is the minimum vertex degree of the graph.

In the next section, we describe basic definitions and notations used in this paper. Section 3 shows the equivalence between the class of CACGs and that of CPGs. In Section 4, we describe how to find in polynomial time, a maximum clique and a maximum independent set of a CACG. Section 5 concludes the paper.

2 Definitions and Notations

Let $G = (V, E)$ be a graph. For two vertices x and $y \in E$ we say that x is *adjacent* to y and vice versa, if $(x, y) \in E$; otherwise they are said to be *independent.* A set of vertices $V' \subseteq V$ is called a *clique* if the vertices in V' are pairwise adjacent. It is called an *independent set* if the vertices in V' are pairwise independent. A *maximum clique* (resp., *maximum independent set*) of G is one with the maximum cardinality

among all cliques (resp., independent sets) of G.

Let $I = \{I_1, I_2, \ldots, I_n\}$ be a family of intervals on the real line. The endpoints of each interval in I are assigned their coordinates on the real line. We denote by l_i and r_j the coordinates of the left and right endpoints, respectively, of I_j, that is, $I_j = [l_j, r_j]$. For every two intervals $I_j, I_k \in I$, we say that I_j *contains* I_k, denoted by $I_k \subset I_j$, if $l_j \leq l_k \leq r_k \leq r_j$. If neither of them contains the other, I_j and I_k are said to be *independent* of each other. A subfamily $I' = \{I_{j_1}, \ldots, I_{j_k}\} \subseteq I$ is said to be *complete* if, for every pair of intervals I_{j_i} and I_{j_l}, $1 \leq i, l \leq k$, either $I_{j_i} \subset I_{j_l}$ or $I_{j_l} \subset I_{j_i}$. I' is said to be *independent* if its intervals are pairwise independent. A *maximum complete* (resp., *maximum independent*) *interval family* is one with the maximum cardinality among all complete (resp., independent) subfamilies of I. The vertices corresponding to the intervals in a complete (resp., independent) subfamily of I form a clique (resp., independent set) of the ICG G_I.

Let $A = \{A_1, A_2, \ldots, A_n\}$ be a family of n arcs on a circle. The endpoints of each arc in A are assigned their coordinates on the circumference of the circle. Among them, the one which is encountered earlier while traversing the arc in the clockwise direction is called the *head* and the other is called the *tail.* Each arc A_j is thus uniquely represented as $A_j = [h_j, t_j]$, where h_j and t_j are the coordinates of the head and tail, respectively. For two arcs $A_j = [h_j, t_j]$ and $A_k = [h_k, t_k]$, we say that A_j *contains* A_k, denoted by $A_k \subset A_j$, if points h_j, h_k, t_k, t_j are encountered in this order while traversing the circle in the clockwise direction. If neither of them contains the other, A_j and A_k are said to be *independent* of each other. A subfamily $A' = \{A_{j_1}, \ldots, A_{j_k}\} \subseteq A$ is said to be *complete* if, for every pair of arcs A_{j_i} and A_{j_l}, $1 \leq i, l \leq k$, either $A_{j_i} \subset A_{j_l}$ or $A_{j_l} \subset A_{j_i}$. A' is said to be *independent* if its arcs are pairwise independent. A *maximum complete* (resp., *maximum independent*) *arc family* is one with the maximum cardinality among all complete (resp., independent) subfamilies of A. The vertices

corresponding to the arcs in a complete (resp., independent) subfamily of A form a clique (resp., independent set) of the CACG G_A.

A family of arcs A is said to be *canonical* if

1. All endpoints of the arcs in A are assigned distinct integer coordinates between 1 and $2n$,

2. No single arc covers the circle by itself, and

3. The coordinate assigned to the head of arc A_1 is the positive integer 1 and all the other endpoints are assigned coordinates in ascending order in the clockwise direction.

If A is not canonical, it can be converted in $O(nlogn)$ time into a canonical family A' such that $G_{A'} = G_A$ [10]. In a canonical family of arcs, an arc $A_j = [h_j, t_j]$ is called a *forward arc* if $h_j < t_j$; otherwise it is said to be a *backward arc.*

Let π be a permutation of numbers $1, 2, \ldots, n$ and let C_{in} and C_{out} be two concentric circles, such that C_{in} is smaller in diameter than C_{out}. Let n points be labeled on C_{in} by $1', 2', \ldots, n'$ and on C_{out} by $\pi(1), \pi(2), \ldots, \pi(n)$, respectively, in the clockwise direction. A *circular permutation diagram* (abbreviated as CPD) consists of the labeled circles C_{in} and C_{out} and a set of chords $\{W_1, W_2, \ldots, W_n\}$, such that W_i joins point i' on C_{in} to point i on C_{out}, $1 \leq i \leq n$. These chords are drawn in the annular region between C_{in} and C_{out} in such a way that any two of them intersect at most once. A circular permutation graph is an intersection graph for the set of chords in a CPD. Fig. 1 shows an example of a CPD and its corresponding CPG. If the direction in the traversal of a chord in a CPD from one endpoint on C_{in} to the other endpoint on C_{out} is clockwise (resp., counterclockwise), it is called a *clockwise chord* (resp., *counterclockwise chord*). For example, the CPD of Fig. 1(a) has three

clockwise chords, W_1, W_2 and W_3. Note that a chord whose endpoints on C_{in} and C_{out} fall on the same radial line is regarded as a clockwise chord for convenience.

3 Equivalence of a CPG and a CACG

In this section we establish the equivalence of a CACG and a CPG. It suffices to show that for every canonical family $A = \{A_1, A_2, \ldots, A_n\}$ of arcs on a circle C, there exists a CPD $D = \{W_1, W_2, \ldots, W_n\}$ and vice versa, such that for every pair of arcs A_i and A_j, one of which contains the other, the corresponding chords W_i and W_j intersect each other, and vice versa. We say that such A and D are *functionally equivalent.*

3.1 Construction of a Functionally Equivalent CPD from a Family of Arcs

The following procedure constructs a CPD D from A in $O(n)$ time. See Fig. 2.

Procedure 1 :

1. Draw two circles C_{in} and C_{out} concentric with the circle C. Both C_{in} and C_{out} are smaller in diameter than C, with C_{in} having a smaller diameter than C_{out}.
2. For every arc $A_j = [h_j, t_j]$ project h_j radially onto C_{in} and call the point on it p_j. Project t_j radially onto C_{out} and call the point on it q_j.
3. In the annular region between C_{in} and C_{out} draw a chord W_j in the clockwise direction, joining p_j with q_j. □

It is easy to see that the CPD D thus constructed and A are functionally equivalent, and hence the CPG for D is exactly identical to the CACG G_A. Thus we have

the following lemma :

Lemma 1 *Every CACG is a CPG.* □

3.2 Construction of a Functionally Equivalent Family of Arcs from a CPD

The CPD obtained from the family of arcs by Procedure 1 consists of clockwise chords only. We call a CPD with all the chords oriented in the same direction (either clockwise or counterclockwise), a *monosense* CPD. A chord in a CPD that spans more than 2π radians and hence appears to overlap itself is called an *XL-chord.* For example W_2 in Fig. 1(a) is an XL-chord. A CPD without XL-chords is called a *proper* CPD.

Let D be a CPD with chords arbitrarily oriented in both directions. The following two steps transform D into a proper monosense CPD D_{pm}, such that the CPGs for D and D_{pm} are identical. The first step converts D to a monosense CPD D_m. In the second step we convert D_m into a proper monosense CPD D_{pm}.

Step 1 : Construction of a monosense CPD

Let W_j be a counterclockwise chord in D that spans the largest angle. Let p_j and q_j be its endpoints on C_{in} and C_{out} respectively. Keeping the endpoints on C_{in} in their positions, shift all the endpoints on C_{out} in the clockwise direction until q_j passes p_j radially. We call this intermediate diagram D_m. Fig. 3 illustrates an example of the execution of this step.

Effect of Step 1

At the end of Step 1, all counterclockwise chords become clockwise, and thus

D_m is monosense. Although some chords may have been converted into XL-chords, none of the intersections were affected and no two chords intersect at more than one point. Suppose that D_m has an XL-chord W_{XL}. As shown in Fig. 4, this chord creates a region between its two endpoints p_{XL} and q_{XL} where the chord essentially appears to overlap itself. We denote this overalp region by OV. No other chord is completely contained in OV, since if it were, it would intersect W_{XL} twice, which is not legitimate. Thus there are four kinds of chords relative to W_{XL} as shown in Fig. 4. Chords such as W_i have their tails inside the region OV but the heads outside it. Chords such as W_k have their heads inside OV but their tails outside it. A third kind of chords such as W_o have both endpoints outside the region OV. A fourth kind of chords such as W_s have both endpoints inside OV but are not completely contained inside OV.

Step 2 : Construction of a proper monosense CPD

For each XL-chord W_{XL}, slide q_{XL} in the counterclockwise direction, along with all other points on the circumference of C_{out} in the region OV so that q_{XL} just gets past p_{XL} radially. Fig. 5 shows the result of this shrinking process of the XL-chord of Fig. 4.

Effect of Step 2

Step 2 only shrinks some chords and creates no counterclockwise chords. Chords such as W_k and W_o are unaffected. Chords such as W_i and W_s only shrink but remain clockwise if their endpoints on C_{out} are kept close enough to the new position of q_{XL}. We show that given two XL-chords which are to be shrunk, the order in which the shrinking is done is entirely inconsequential. Let B_1 and B_2 be two arbitrary XL-chords with their respective overlap regions OV_1 and OV_2. The following three possibilities arise :

1. OV_1 is contained in OV_2.

2. OV_1 and OV_2 are disjoint.

3. OV_1 and OV_2 have a nonempty intersection region.

As is evident from Figs. 6, 7, and 8, the order in which two XL-chords are shrunk is not consequential in all these cases. Therefore, by shrinking all the XL-chords, one by one, we obtain a proper monosense CPD D_{pm}.

The application of Procedure 1 in reverse order to the proper monosense CPD D_{pm} always yields a functionally equivalent family of arcs. Thus, we have established the following lemma :

Lemma 2 *Every CPG is a CACG.* □

Lemmas 1 and 2 imply the following:

Theorem 1 *The class of CACGs is equivalent to the class of CPGs.* □

In the above construction, Step 1 takes $O(n)$ time, since it involves shifting the endpoints on C_{out} of all chords . Step 2 needs $O(n)$ time to shrink each XL-chord, and there are at most n XL-chords. Thus it requires a total of $O(n^2)$ time. Finally, Procedure 1 in reverse order takes $O(n)$ time to execute. Therefore, the construction of a functionally equivalent family of arcs on a circle from a CPD can be done in $O(n^2)$ time.

4 Algorithms for Circular-Arc Containment Graphs

Given a canonical family of arcs $A = \{A_1, A_2, \ldots, A_n\}$, we develop here efficient algorithms for finding a maximum clique and a maximum independent set of the CACG G_A.

4.1 Maximum Clique of a CACG

We transform A into a family of intervals I_A in the following manner :

1. For each backward arc $A_j = [h_j, t_j]$, where $h_j > t_j$, create an interval $I_j^{bk} = [h_j, t_j + 2n]$ of type *back*.

2. For each forward arc $A_j = [h_j, t_j]$, where $h_j < t_j$, create two intervals $I_j^{fw} = [h_j, t_j]$ and $I_j^{dp} = [h_j + 2n, t_j + 2n]$, of types fwd and *dup*, respectively.

Thus, if there are b backward arcs and f forward arcs in A ($n = b + f$), we have a total of $2f + b \leq 2n$ intervals in I_A.

Lemma 3 *For any complete subfamily A' of A, I_A has a complete subfamily of the same cardinality.*

Proof : Let $A' = \{A_{j_1}, A_{j_2}, \ldots, A_{j_k}\}$ with $A_{j_1} \subset A_{j_2} \subset \ldots \subset A_{j_k}$. If A' consists of forward arcs only, both $\{I_{j_1}^{fw}, I_{j_2}^{fw}, \ldots, I_{j_k}^{fw}\}$ and $\{I_{j_1}^{dp}, I_{j_2}^{dp}, \ldots, I_{j_k}^{dp}\}$ are complete in I_A. If A' consists of backward arcs only, then $\{I_{j_1}^{bk}, I_{j_2}^{bk}, \ldots, I_{j_k}^{bk}\}$ is complete. Suppose that A' consists of s forward arcs and $k - s$ backward arcs. Since no forward arc contains a backward arc, it is clear that $A_{j_1}, A_{j_2}, \ldots, A_{j_s}$ are forward arcs and $A_{j_{s+1}}, A_{j_{s+2}}, \ldots, A_{j_k}$ are backward arcs. In this case, either $\{I_{j_1}^{fw}, I_{j_2}^{fw}, \ldots, I_{j_s}^{fw}, I_{j_{s+1}}^{bk}, I_{j_{s+2}}^{bk}, \ldots, I_{j_k}^{bk}\}$ or $\{I_{j_1}^{dp}, I_{j_2}^{dp}, \ldots, I_{j_s}^{dp}, I_{j_{s+1}}^{bk}, I_{j_{s+2}}^{bk}, \ldots, I_{j_k}^{bk}\}$ is complete. Thus in any case, I_A has a complete subfamily of cardinality k. □

Let $I_A{}'$ be any complete subfamily of I_A. It is clear that the subfamily of A corresponding to $I_A{}'$ is complete. Furthermore, its cardinality is equal to $|\ I_A{}'\ |$ since $I_A{}'$ can not contain both types fwd and *dup* at the same time. Thus by Lemma 3 we have the following theorem :

Theorem 2 *For any maximum complete interval family of I_A, the corresponding subfamily of A is a maximum complete arc family of A.* □

The conversion of A into I_A takes $O(n)$ time. Given a family I of m intervals, a maximum complete interval family of I can be obtained in $O(mlogm)$ time [9]. Thus by Theorem 2, we can compute a maximum clique of the CACG G_A in $O(nlogn)$ time.

4.2 Maximum Independent Set of a CACG

For each arc $A_j \in A$ let Y_{A_j} be a subfamily of maximum cardinality among all independent subfamilies of A that contain A_j. Those arcs that are contained in A_j and those that contain A_j can clearly be disregarded in the computation of Y_{A_j}. All the other arcs can be classified into the following categories (see Fig. 9) :

1. Arcs such as A_i where only the head h_i is contained in A_j.
2. Arcs such as A_k where only the tail t_k is contained in A_j.
3. Arcs such as A_d which are entirely disjoint from A_j.
4. Arcs such as A_s where the head h_s and the tail t_s are both contained in A_j but the arcs themselves are not contained in A_j.

After those arcs that are not independent of A_j are removed, the circle can be *split open*, in $O(n)$ time, by the steps outlined below, transforming each of the remaining arcs into an interval so that :

1. The tail of arc A_j becomes the starting reference, and the head of A_j becomes the ending reference.

2. All arcs such as A_i are opened with their heads to the left of the starting reference and all arcs such as A_k are opened with their tails to the right of the ending reference. Arcs such as A_d appear between the reference marks while arcs such as A_s are stretched out with their heads to the left of the starting reference and their tails to the right of the ending reference. See Fig. 10.

We denote the resultant family of intervals by I^j. An $O(mlogm)$ algorithm for finding a maximum independent family of a family of m intervals is known [9]. Therefore we can obtain a maximum independent interval family of I^j in $O(nlogn)$ time. Let X^j be the family of arcs in A that corresponds to such an independent interval family. Since every arc which corresponds to an interval in I^j is independent of A_j, we can determine Y_{A_j} to be $X^j \cup \{A_j\}$.

Let v_α be a vertex of G_A with the minimum degree δ. Let A_α be the arc corresponding to v_α. As we have shown above, we can find Y_{A_α} in $O(nlogn)$ time. Let Y denote the maximum one among all independent arc families of A that do not contain A_α. Y must contain at least one of the δ arcs that are not independent of A_α. For each of these arcs $A_{j_1}, A_{j_2}, \ldots, A_{j_k}, \ldots, A_{j_\delta}$, we can find $Y_{A_{j_k}}$ in $O(nlogn)$ time by splitting open the circle as shown earlier. A maximum independent arc family of A is then given by Y_{A_γ} where $\gamma \in \{\alpha, j_1, \ldots, j_\delta\}$ and $| Y_{A_\gamma} | = \max \{ | Y_{A_\alpha} |, | Y_{A_{j_1}} |, \ldots, | Y_{A_{j_\delta}} | \}$.

The above approach thus takes a total of $O(\delta nlogn)$ time. From the given family of arcs A, it takes $O(nlogn + e)$ time to obtain G_A and determine a minimum degree vertex v_α [10]. Thus the entire maximum independent arc family algorithm runs in $O(\delta nlogn + e)$ time.

5 Conclusion

We have introduced a new class of containment graphs called circular-arc containment graphs and presented two efficient algorithms for them. We have established the equivalence between the class of circular permutation graphs and the class of circular-arc containment graphs by means of a constructive proof. Thus, the proposed algorithms are applicable to circular permutation graphs as well. Furthermore, using the recognition algorithm for a circular permutation graph developed by Rotem and Urrutia [12], we can test in $O(n^{2.49+})$ time whether a graph with n vertices is a circular-arc containment graph.

References

[1] C. J. Colbourn, "On Testing Isomorphism of Permutation Graphs," *Networks*, Vol. 11, pp. 13-21, 1981.

[2] B. Dushnik and E. W. Miller, "Partially Ordered Sets," *Amer. J. Math.*, Vol. 63, pp. 600-610, 1941.

[3] S. Even, *Graph Algorithms*, Computer Science Press, Potomac, MD, 1979.

[4] S. Even, A. Pnueli, and A. Lempel, "Permutation Graphs and Transitive Graphs," *J. Assoc. Comput. Mach.*, Vol. 19, pp. 400-410, 1972.

[5] M. Farber and J. M. Keil, "Domination in Permutation Graphs," *J. Algorithms*, Vol. 6, pp. 309-321, 1985.

[6] M. L. Fredman, " On Computing the Length of Longest Increasing Subsequences," *Discrete Math.*, Vol. 11, pp. 29-35, 1975.

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

[8] M. C. Golumbic, "Containment Graphs and Intersection Graphs," Technical Report 135, IBM Israel Scientific Center, Haifa, Israel, 1984.

[9] S. Masuda and K. Nakajima , " Optimal Algorithms for Interval Containment Graphs," *Proc. 29th Midwest Symposium on Circuits and Systems,* Lincoln, NE, August 1986, pp. 431-434.

[10] M. V. Nirkhe, " Efficient Algorithms for Circular-Arc Containment Graphs," Technical Report SRC TR 87-211, Systems Research Center, University of Maryland, College Park, MD, 1987.

[11] A. Pnueli, A. Lempel and S. Even, "Transitive Orientation of Graphs and Identification of Permutation Graphs," *Can. J. Math.*, Vol. XXIII, pp. 160-175, 1971.

[12] D. Rotem and J. Urrutia, "Circular Permutation Graphs," *Networks*, Vol. 12, pp. 429-437, 1982.

[13] J. Spinard, " On Comparability and Permutation Graphs," *SIAM J. Comput.*, Vol. 14, pp. 658-670, 1985.

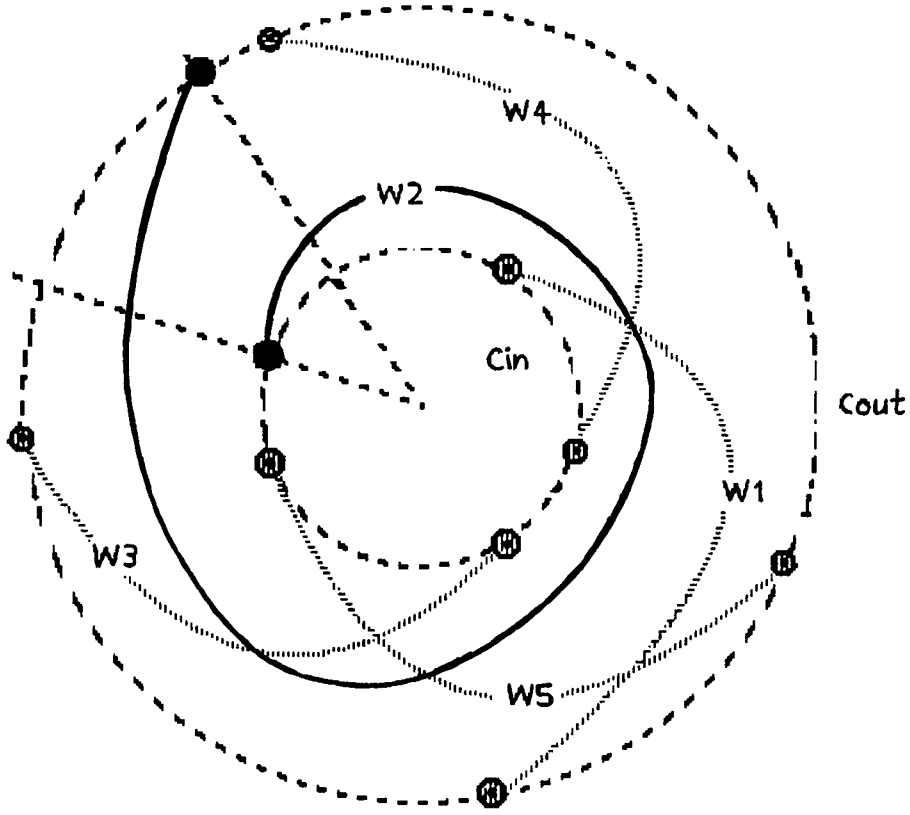

Fig. 1(a). A circular permutation diagram.

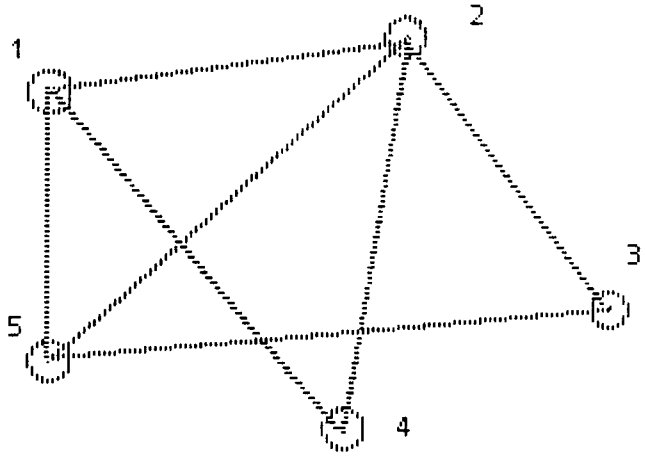

Fig. 1(b). The CPG for the CPD of Fig. 1(a).

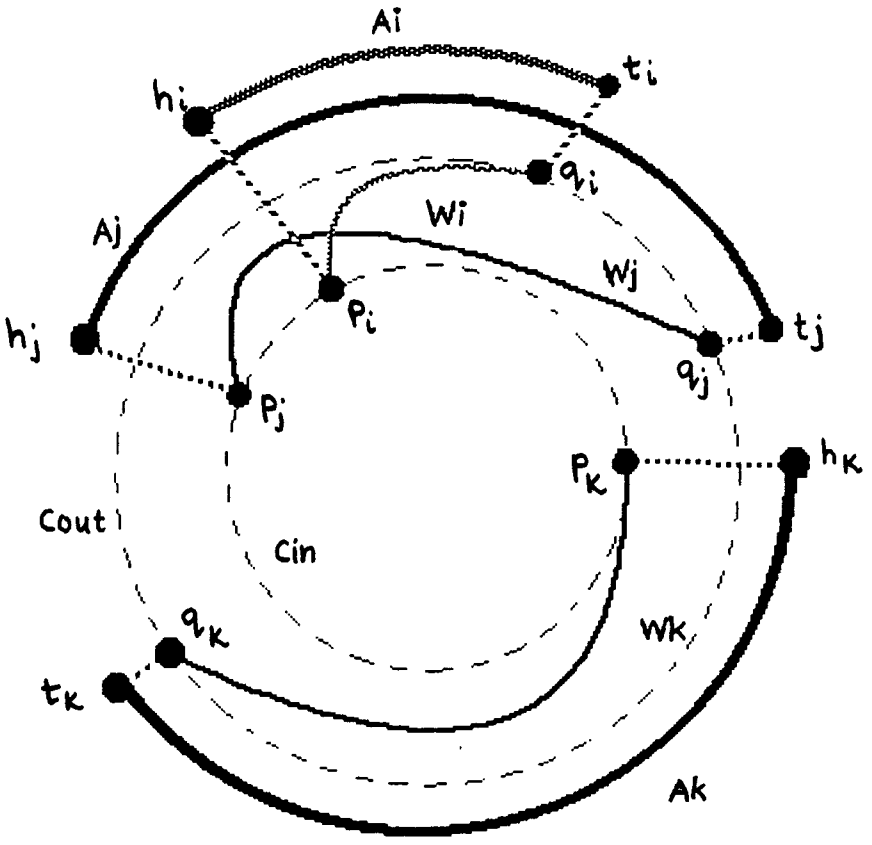

Fig. 2. Construction of a CPD from a family of arcs.

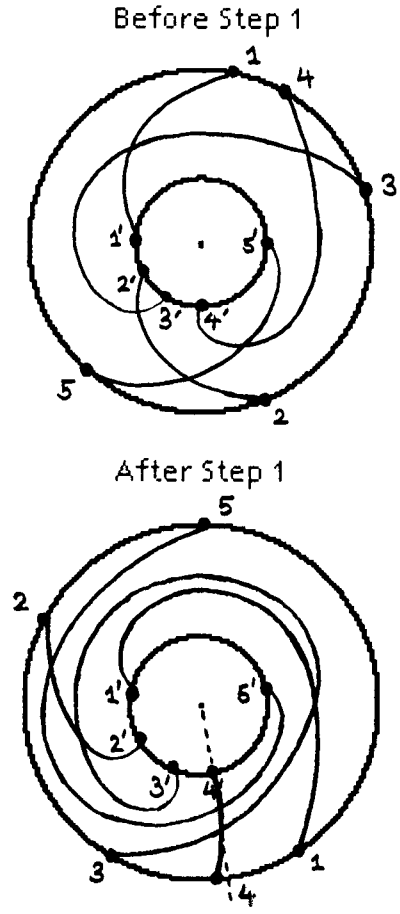

Fig. 3. Construction of a monosense CPD.

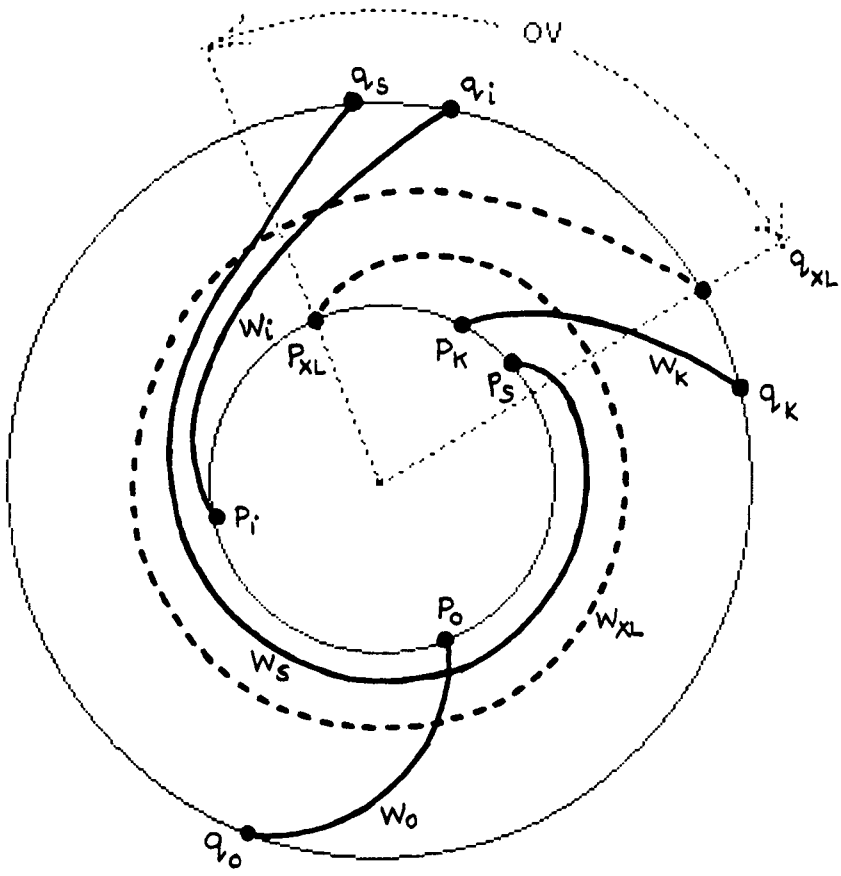

Fig. 4. An XL-chord and other chords relative to it.

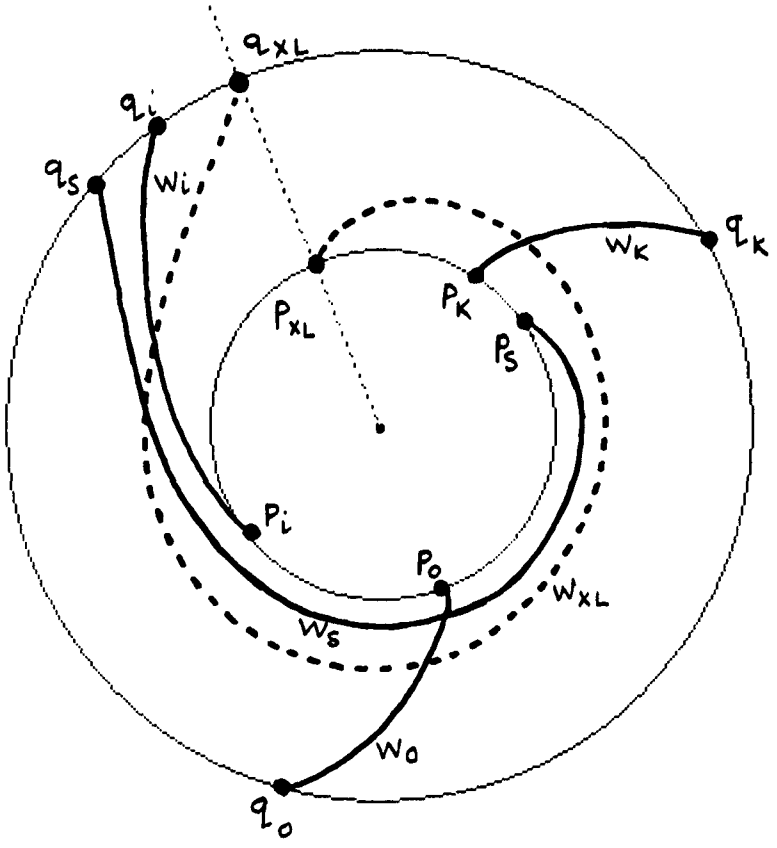

Fig. 5. 'Shrinking' of the XL-chord of Fig. 4.

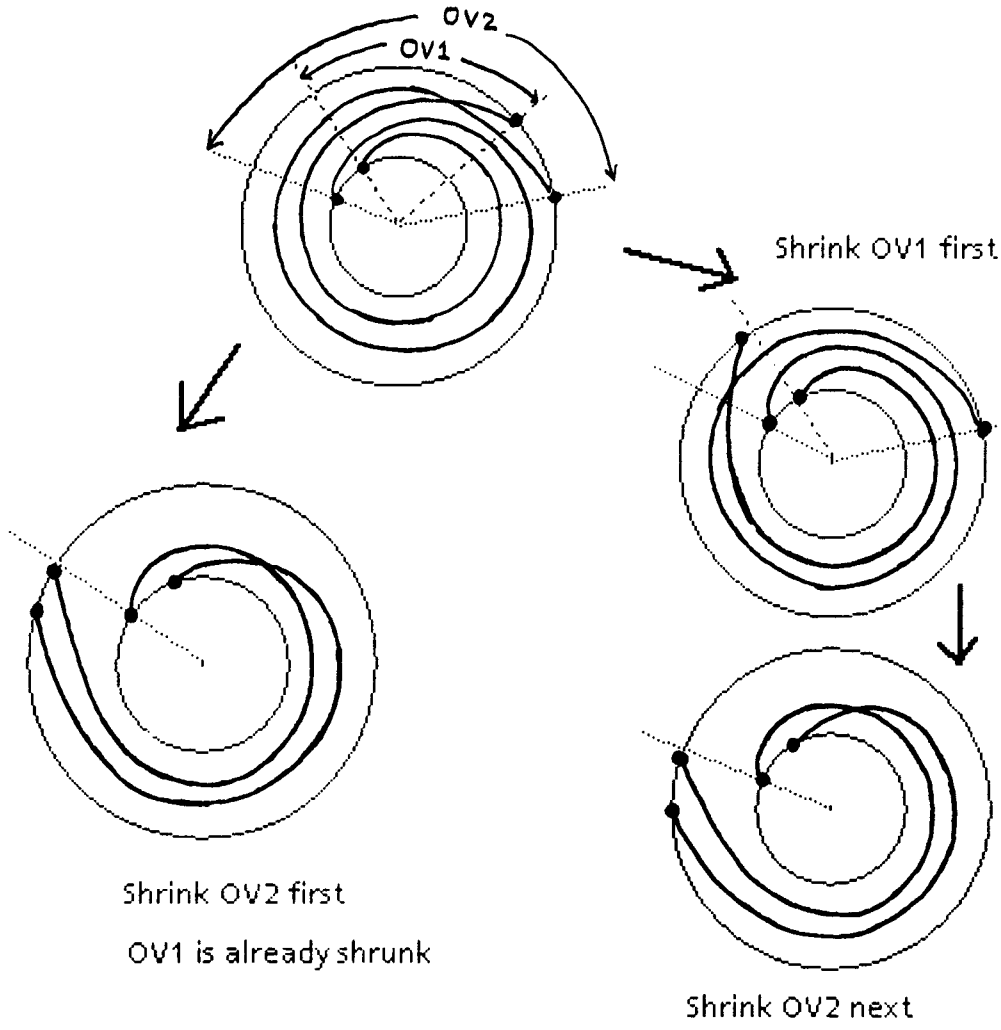

Fig. 6. OV_1 is contained in OV_2.

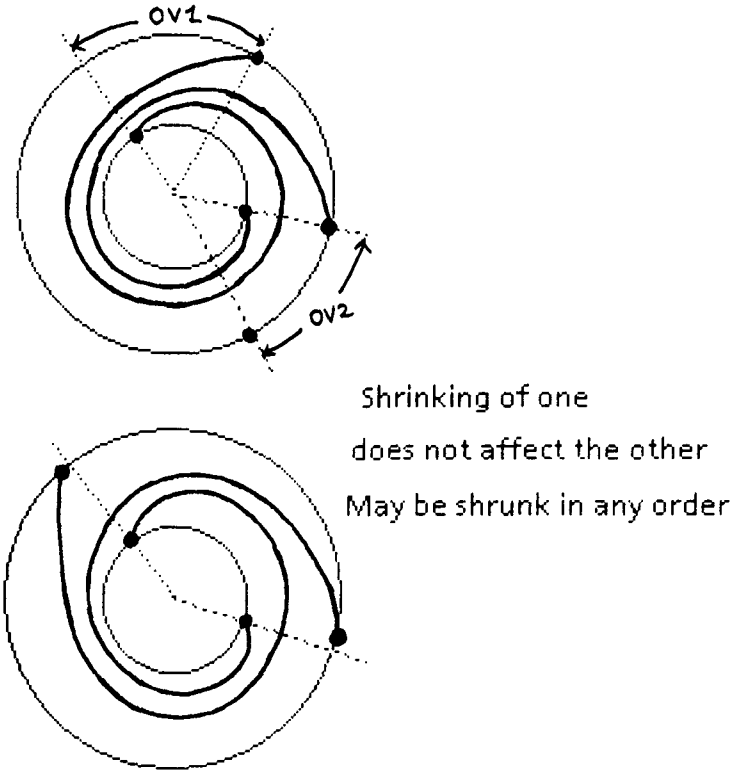

Fig. 7. OV_1 **and** OV_2 **are disjoint.**

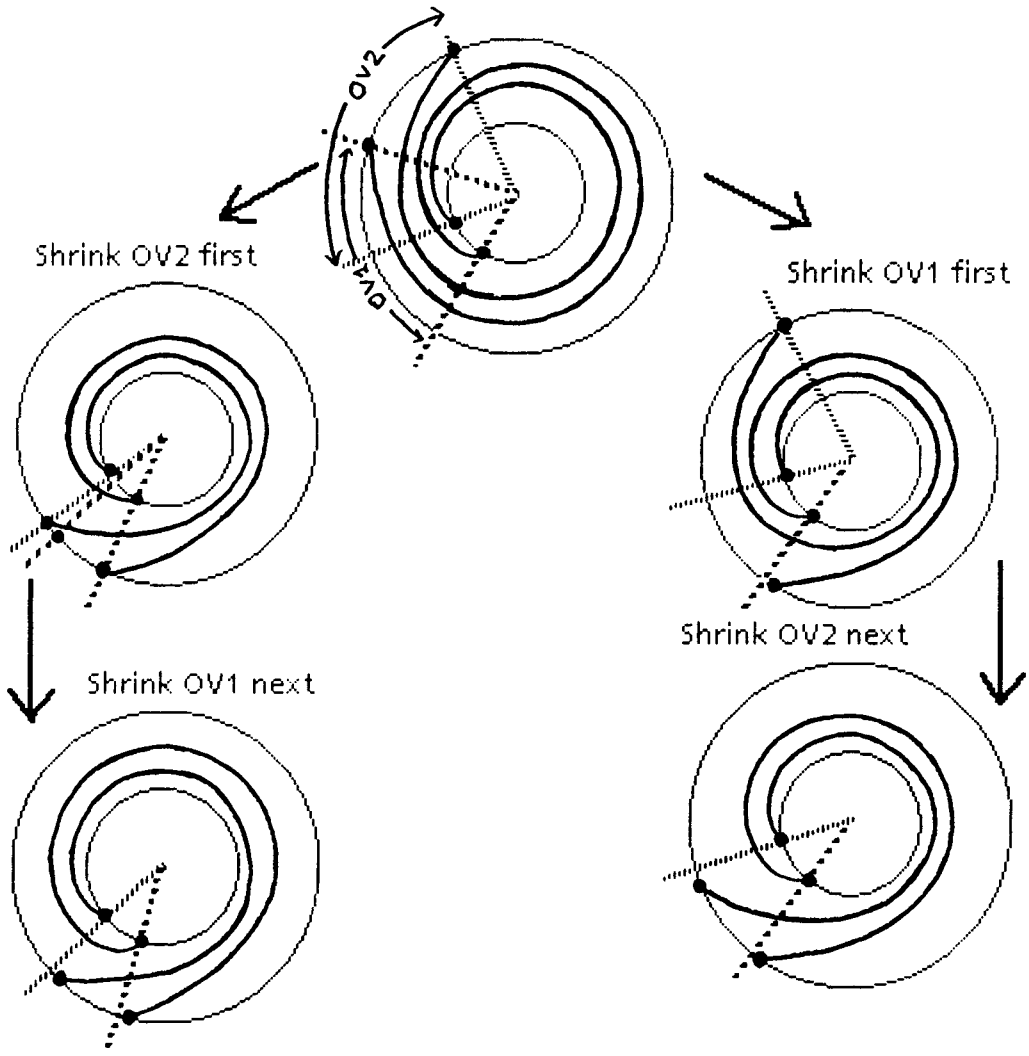

Fig. 8. OV_1 **and** OV_2 **have a nonempty intersection.**

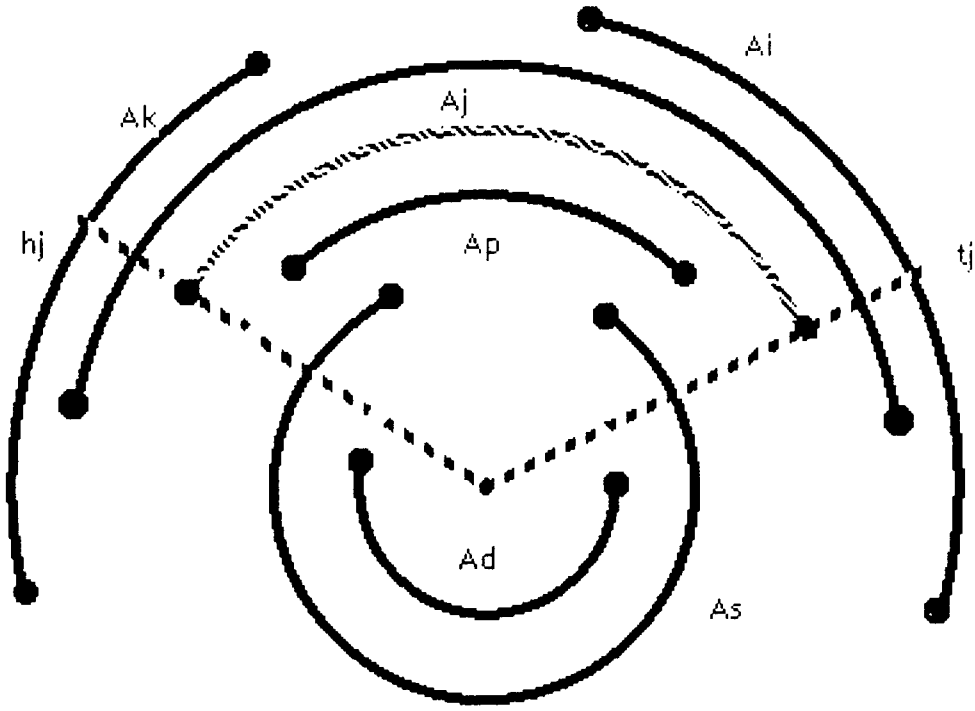

Fig. 9. An arc A_j and other arcs with respect to it.

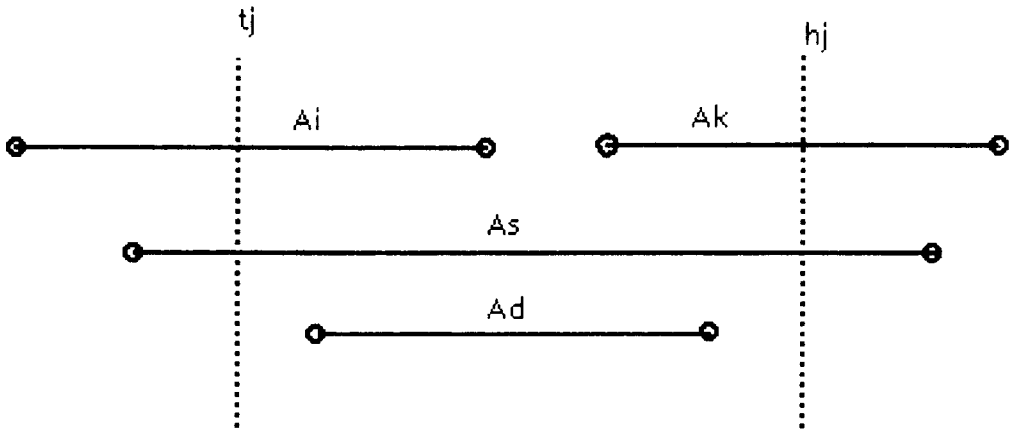

Fig. 10. Splitting the circle open.