

サークルグラフの最大重み独立集合を求める  
より高速なアルゴリズム

浅野孝夫\*, 今井 浩\*\*, 向山 輝\*

\*上智大学理工学部機械工学科    \*\*九州大学工学部情報工学科

サークルグラフの最大重み独立集合を求めるより高速なアルゴリズムを提案する。N個の端点からなるn本の弦のサークルグラフに対して、そのアルゴリズムは、最大重み独立集合を $O(nN)$ 時間、 $O(n)$ 領域で求める。

A Faster Algorithm for  
Finding a Maximum Weight Independent Set of a Circle Graph

Takao ASANO\*, Hiroshi IMAI§ and Akira MUKAIYAMA\*

\*Department of Mechanical Engineering, Sophia University  
Chiyoda-ku, Tokyo 102, Japan

§Department of Computer Science and Communication Engineering  
Kyushu University, Fukuoka 812, Japan

We present an algorithm for finding a maximum weight independent set of a circle graph. For a circle graph of a set of n chords with N endpoints, the algorithm finds a maximum weight independent set in  $O(nN)$  time and  $O(n)$  space.

## 1. Introduction

The problem of decomposing a polygon into simpler components is of fundamental importance in computational geometry and often arises in VLSI layouts. Closely related to this problem is the problem of finding a maximum set of independent chords in a circle (i.e., the problem of finding a maximum independent set of a circle graph). In fact, algorithms for finding a maximum set of independent chords in a circle are used in decomposing a simple polygon into a minimum number of trapezoids [AAI] and into a minimum number of uniformly monotone polygons [LN].

Gavril first obtained an  $O(n^3)$  time algorithm for finding a maximum set of independent chords for a set of  $n$  chords with  $N$  endpoints on a circle. Later, Read et al. [RRUI], Buckingham [B], and Asano et al. [AAI] proposed  $O(n^2)$  time algorithms. Recently, Liu and Ntafos proposed an  $O(N^3)$  time and  $O(N^2)$  space algorithm [LN]. Note that  $n=O(N^2)$  (and  $n^2=O(N^4)$ ) in the worst case.

In this paper, we present another efficient algorithm which is a slight modification of the algorithm in [AAI]. The proposed algorithm runs in  $O(nN)$  time and  $O(n)$  space. Thus, it may be considered to be an improvement of  $O(n^2)$  time algorithms as well as Liu and Ntafos's algorithm.

## 2. Preliminaries

For a set of chords each of which connects two points on a circle  $C$ , a subset of chords is independent if no two of them have a point in common. Here, we consider each chord is a closed set, i.e., both of its endpoints on  $C$  belong to it (the argument below can be easily modified in case each chord is an open set). Then the problem we consider is to find a maximum set of independent chords in  $C$ . This can be restated as a problem on special graphs known as circle graphs, intersection graphs of chords in circles. That is, in a circle graph, each vertex represents a chord in a circle  $C$  and two vertices are connected by an edge iff the corresponding chords intersect (i.e., have a point in common). Thus, finding a maximum set of independent chords is equivalent to finding a maximum independent set of a circle graph (an independent set of a graph is a subset of vertices no two of which are connected by an edge). When each vertex has a weight, the weight of an independent set is defined to be the sum of the weights of vertices in the set.

A circle graph is closely related to an overlap graph defined below. Let  $S$  be a set of  $n$  intervals on a line with  $N$  endpoints. For intervals  $I=[x^-,x^+]$  and  $J=[y^-,y^+]$ , we say  $I$  is completely contained in  $J$  if  $y^- < x^-$  and  $x^+ < y^+$ . Two intervals are said to overlap if they intersect without one being completely contained in the other. An overlap graph  $G(S)$  of  $S$  is obtained by identifying each interval with a vertex and connecting two vertices by an edge iff the corresponding intervals overlap. An interval graph, an intersection graph of intervals, is obtained by identifying each

interval with a vertex and connecting two vertices by an edge iff the corresponding intervals intersect. There is a slight difference between an overlap graph and an interval graph. Since the class of circle graphs coincides with the class of overlap graphs [G], we will concentrate on finding a maximum weight independent set of an overlap graph.

### 3. Finding a maximum weight independent set of an overlap graph

Consider a set  $S$  of  $n$  intervals with  $N$  endpoints. Each interval  $I$  of  $S$  has a nonnegative weight  $w(I)$ . In this section we will describe an algorithm for finding a maximum weight independent set of the overlap graph  $G(S)$  of  $S$ . We assume that  $N$  endpoints are sorted and labeled from 1 to  $N$ . This can be done in  $O(N \log N)$  time and  $O(N)$  space (if the set of endpoints is considered to be a multiset, this sorting can be done in  $O(n \log n)$  time and  $O(n)$  space). We ignore this complexity because it does not dominate the complexity of our algorithm (note that  $N = \Omega(\sqrt{n})$ ). For convenience sake, we introduce a new interval  $I_0 = [0, N+1]$  with weight  $w(I_0) = 0$ . All intervals of  $S$  are of course completely contained in  $I_0$ .

For an interval  $I$  of  $S$ , let  $S(I)$  be the set of intervals of  $S$  completely contained in  $I$ . Define  $W(I)$  to be the weight of a maximum weight independent set of the overlap graph of  $S(I) \cup \{I\}$  (that always contains  $I$  if  $w(I)$  is positive). Thus, it is easily observed that  $W(I)$  satisfies the following [G], [AAI], [IA]:

$$W(I) = w(I) + \left[ \text{the weight of a maximum weight independent set of the interval graph of } S(I) \text{ where the weight of each interval } J \text{ of } S(I) \text{ is } W(J) \right]$$

Thus,  $W(I_0)$  becomes the weight of a maximum weight independent set of the overlap graph  $G(S)$  of  $S$  (and  $G(S \cup \{I_0\})$ ).

Gavril proposed an  $O(n^3)$  time algorithm based on the above observation [G]. In fact, he has iteratively solved the problem of finding a maximum weight independent set of an interval graph for a subset of those intervals to compute  $W(I)$ 's. Note that, to compute  $W(I)$ , we have only to compute  $W(J)$  in advance for each  $J$  of  $S$  completely contained in  $I$ . Thus, there are many possibilities about the order to compute  $W(I)$ 's. In this paper we adopt a method of computing  $W(I)$ 's in decreasing order of the left endpoints of the intervals. Note also that, for intervals  $I$ 's having their left endpoint in common, we can compute  $W(I)$ 's simultaneously.

Let  $L_k$  ( $k=0, 1, \dots, N-1$ ) be the set of intervals of  $S \cup \{I_0\}$  having  $k$  as their left endpoint. Let  $I_k$  be the interval in  $L_k$  with the largest right endpoint.  $S(I_k)$  is the set of intervals of  $S$  completely contained in  $I_k$ . Let  $\{x_1, \dots, x_{a(k)}\}$  be the set of endpoints of the intervals in  $L_k \cup S(I_k)$ . We assume  $x_1 < \dots < x_{a(k)}$ . Then our algorithm for computing  $W(I)$ 's ( $I \in L_k$ ) can be described as follows (we assume that, for each  $J$  in  $S(I_k)$ ,  $W(J)$  is already computed).

Procedure COMP\_W( $L_k, I_k, S(I_k)$ );

begin

```
1  sort a(k) numbers  $x_1, \dots, x_{a(k)}$  in increasing order;
   {comment: we assume  $x_1 < \dots < x_{a(k)}$ .  $x_1$  is the left endpoint of the
   intervals in  $L_k$ .}
2  U:=0;
3  for i:=2 to a(k) do
   begin
4     for each interval J in  $S(I_k)$  with  $x_i$  as its left endpoint do
       {comment: U is the weight of a maximum weight independent
       set of the interval graph of intervals in  $S(I_k)$  whose right
       endpoints are less than  $x_i$ }
5     U(J):= U + W(J);
6     if  $x_i$  is the right endpoint of an interval I in  $L_k$  then
7     W(I):= U + w(I);
8     UMAX:= 0;
9     for each interval J in  $S(I_k)$  with  $x_i$  as its right endpoint do
10    if UMAX < U(J) then
11    UMAX:= U(J);
       {comment: UMAX=max[U(J)]}
12    if UMAX > U then
13    U:= UMAX;
   end;
end;
```

The validity of the algorithm can be easily shown by an argument similar to the one in [AAII] (since, at each  $x_i$ , U is the weight of a maximum weight independent set of the interval graph of those intervals in  $S(I_k)$  whose right endpoints are less than  $x_i$ ). Thus, the whole algorithm for computing  $W(I_0)$ , the weight of a maximum independent set of the overlap graph  $G(S)$  of S, as well as all other  $W(I)$ 's, can be written as follows:

Procedure COMP\_WEIGHT;

begin

```
for k:=N-1 downto 0 do
  if k is the left endpoint of an interval of S then
    begin
      find  $L_k, I_k$  and  $S(I_k)$ ;
      COMP_W( $L_k, I_k, S(I_k)$ );
    end;
end;
```

Now we consider the time complexity. Line 1 in  $\text{COMP\_W}(L_k, I_k, S(I_k))$  can be done in  $O(N)$  time since the endpoints are labeled from 1 to  $N$ . Lines 3-13 can be done in  $O(|L_k| + |S(I_k)|)$  time. Thus,  $\text{COMP\_W}(L_k, I_k, S(I_k))$  requires only  $O(n)$  time. Since  $L_k$ ,  $I_k$  and  $S(I_k)$  can be found in  $O(n)$  time,  $\text{COMP\_WEIGHT}$  requires only  $O(nN)$  time. Similarly, it can be shown that the required space is  $O(n)$ . It is also easy to modify the above algorithm in such a way that it actually finds a maximum weight independent set. Thus by summarizing we have:

Theorem. A maximum weight independent set of an overlap graph of  $n$  intervals with  $N$  endpoints (or, of a circle graph of  $n$  chords with  $N$  endpoints on a circle) can be found in  $O(nN)$  time and  $O(n)$  space.

#### 4. Remarks

In [IA], an  $O(n \log n + m_c \log \log n)$  time algorithm for finding a maximum weight independent set of an overlap graph of  $n$  intervals is described, where  $m_c$  is the number of the pairs of intervals one of which is completely contained in the other. Employing the technique described there, we can similarly refine the complexity analysis of our algorithm by using the persistent search tree proposed by Sarnak and Tarjan [ST] (the hive graph proposed by Chazelle [C] can also be used). The persistent search tree is one of the most powerful data structures in computational geometry. It can be used to efficiently solve the following (static) interval search problem: Given a set  $S$  of  $n$  intervals on a line and a query interval, find the set of intervals in  $S$  which are completely contained in the query interval. In fact, by using the persistent search tree, the set of intervals in  $S$  which are completely contained in the query interval can be enumerated in  $O(t + \log n)$  time in such a way that all the endpoints of the enumerated  $t$  intervals are sorted in nondecreasing order. We can construct the persistent search tree in  $O(n \log n)$  time and  $O(n)$  space.

Thus, by choosing  $I_k$  as a query interval, we can find  $S(I_k)$  in  $\text{COMP\_WEIGHT}$  in  $O(|S(I_k)| + \log n)$  time.  $L_k$  and  $I_k$  can also be obtained in  $O(|L_k|)$  time. Thus, the time complexity of  $\text{COMP\_W}(L_k, I_k, S(I_k))$  (i.e., computing  $W(I)$ 's for all  $I$ 's in  $L_k$ ) becomes  $O(\log n + |L_k| + |S(I_k)|)$  since sorting in Line 1 can be done by merging the two sorted lists of endpoints of  $S(I_k)$  and  $L_k$ . From the above discussion, we see that a maximum weight independent set of the overlap graph  $G(S)$  can be found in  $O(n \log n + \sum_{k=0}^N (\log n + |L_k| + |S(I_k)|)) = O(n \log n + m)$  time and  $O(n)$  space, where  $m = \sum_{k=0}^N |S(I_k)|$ . Clearly,  $m \leq m_c$  and  $m \leq nN$ .

It should be noted that the algorithm in this paper can be easily modified even if we consider each interval (chord) is an open set, i.e., it contains none of its endpoints.

### Acknowledgment.

The first author was supported in part by the Grant in Aid for Scientific Research of the Ministry of Education, Science and Culture of Japan under Grant 62550270 (1988).

### References

- [AAI] T. Asano, T. Asano and H. Imai, Partitioning a polygonal region into trapezoids, J. of ACM, 33 (1986), pp.290-312.
- [B] M. Buckingham, Circle Graphs, Ph.D. Dissertation, Courant Institute, Rept. NSO #21, October 1980.
- [C] B. Chazelle, Filtering search: A new approach to query-answering, SIAM J. Computing, 15 (1986), pp.703-724.
- [G] F. Gavril, Algorithms for a maximum clique and a maximum independent set of a circle graph, Networks, 3 (1973), pp.261-273.
- [IA] H. Imai and T. Asano, Applications of the priority search tree to circle graph problems, CAS86-112, IECEJ Technical Reports in Circuits and Systems, 1986.
- [LN] R. Liu and S. Ntafos, On decomposing polygons into uniformly monotone parts, Inform. Process. Lett., 27 (1988), pp.85-88.
- [RRU] R.C. Read, D. Rotem and Urrutia, Orientations of circle graphs, J. Graph Theory, 6 (1982), pp.325-341.
- [ST] N. Sarnak and R.E. Tarjan, Planar point location using persistent search trees, C. of ACM, 29 (1986), pp.669-679.