Properties of the 2-Circle Graph Class

Christopher D. Wedman

Supervisor: M. Keil

Abstract

This paper presents an extension of the circular arc graph class known as the 2-circle class. The outline of a recognition algorithm and a maximum weight clique algorithm are presented for this new class. Properties of the class are also discussed in terms of further extending the results given.

1 Introduction

Let $G = (V, E)$ be a graph where V denotes the vertices in G and E denotes the edges. A *clique* is a graph in which all vertices are adjacent. A *weighted* graph has weights on its vertices. Let n be the size of V and m be the size of E.

A *tree* is a graph that does not contain any cycles, and is further described as *rooted* if the graph can be directed so that all vertices have indegree 1 except for a vertex designated as the *root* which has an indegree of 0. The tree is described as *unrooted* if this is not true.

An intersection graph $G = (V, E)$ is defined as a graph for which an *Intersection model* I exists, where I consists of a collections of sets F, and there is a one-to-one correspondence between every $f \in F$ and every $v \in V$. If $F(x)$ intersects $F(y)$ in I, then $(x, y) \in E$, where $x, y \in V$. The intersection model I is often used as a basis for algorithms on the intersection graph as it presents the graph in a more structured manner.

One example of an intersection graph is called a *Circular Arc* graph. A Circular Arc graph $G = (V, E)$ is defined as having a one-to-one correspondence between V and a set S of arcs in a circle, where two vertices in V are adjacent if the corresponding arcs in S have a non-empty intersection. S is called an intersection model for G. If S is a collection of intervals on a real line, G is an *Interval* graph.

Another example of intersection graphs is the family of intersection graphs of paths in a tree. A discussion of this family is found in [5] where several types of graphs are studied, including directed and undirected paths in rooted and unrooted trees. The *Chordal* graph class is also an intersection graph, in this case the intersection graph of trees in a tree [2].

In a *2-Circle* graph S is defined as the arcs on a directed circle with a directed chord, with the further limitation that no arc may begin or end on the chord. The point where the chord meets the circle is known as the *chord-circle intersection*. Any arcs that pass through the chord are known as *chordal arcs*. This is an extension of the well known Circular Arc class of graphs, and two circular arc graphs can be seen by looking at the intersection model. As can be seen in the example, one circular arc graph can be formed by removing all arcs passing through the chord, and another by removing all arcs whose endpoints are both to the left of the chord.

It is expected that investigation into the properties of the 2-Circle graph class should lead to general properties that apply to other extensions of the Circular Arc graph classes.

This paper will discuss properties found in the 2-Circle graph and present the outline of a recognition algorithm for the graph class. The recognition algorithm depends upon the properties presented, but could be modified for more general use. It also presents an algorithm for finding the maximum weight clique in a 2-Circle graph given the intersection model.

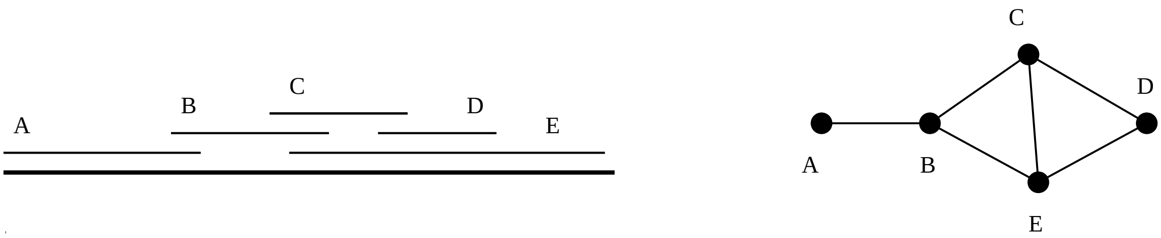

Figure 1: Interval Graph

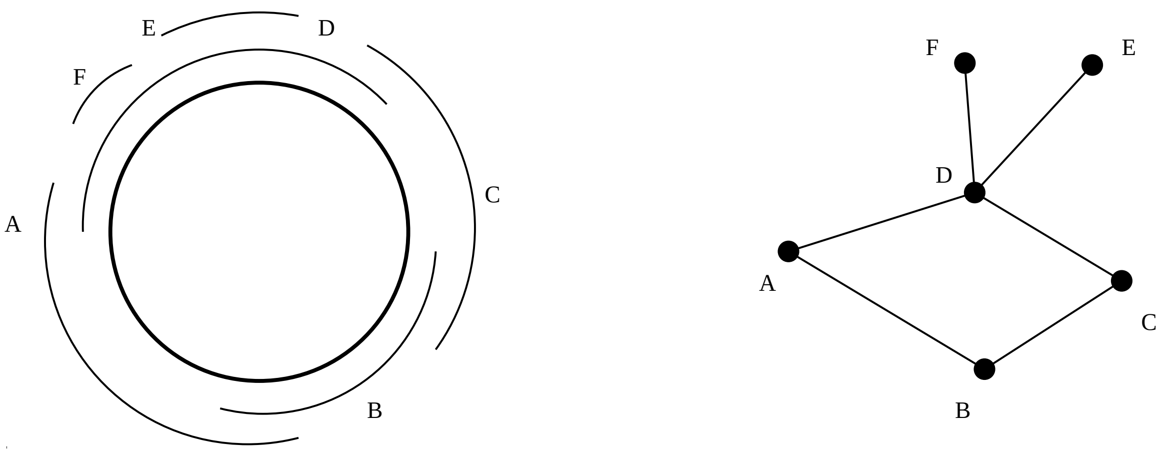

Figure 2: Circular Arc Graph

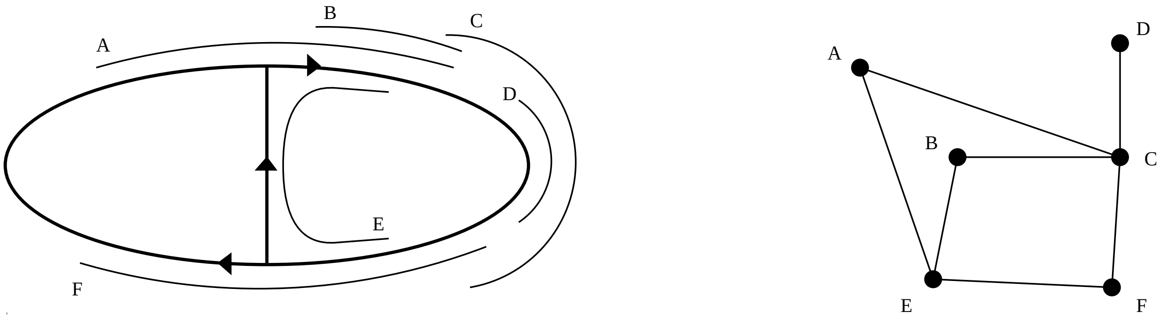

Figure 3: 2-Circle Graph

2 Properties

2.1 Cliques in the 2-Circle Graph

The maximum number of maximal cliques in a 2-circle graph is less than n^2+n, where n is the number of vertices in the graph. There are two types of maximal cliques in a 2-Circle graph, *singular triple* cliques where the arcs in the intersection model form a circle, and *point* cliques where all the arcs in the clique intersect at a common point.

The maximum number of point cliques is the number of arcs or vertices in the graph. This is shown by the fact that if one scans along the collection of arcs, another maximal clique can only be found at the point that another arc begins. If an arc ends, then the new clique at that point is not maximal, as there exists a clique that contains all of the current arcs as well as the arc that just ended. Therefore the maximum number of point cliques is $O(n)$.

The maximum number of singular triple cliques is found by using the fact that in a singular triple, there are three important points. In order to be a member of the clique, an arc must pass through two of the three points on the circle. As will be proved, two of these points can be set to the two endpoints of two of the arcs in the clique. This means that a singular triple can be uniquely identified using two arcs, which implies that the maximum number of singular triple cliques is the number of arcs squared, or $O(n^2)$. Therefore, the maximum number of cliques in the 2-Circle graph is $O(n^2)$.

Given a singular triple clique, there is some arc in the clique whose two endpoints can be used as two of the three points through which all the arcs must pass. As it is possible to move the point until the end of one of arcs, that endpoint can be used as one of the points designating the clique. As an arc need only pass through two of the three points to be a member of the clique, a singular triple clique can be designated using the endpoints of two arcs.

2.2 Head Tail Property

Given an intersection graph, where the vertices in the graph represent paths or arcs in the intersection model, if the intersection of two arcs means that one arc contains the head or tail of the other arc, and may contain both, the graph is said to possess the *Head Tail* property. This property is found in the 2-Circle graph as well as the Circular Arc, Interval, and Directed Path graphs. As shown in the example, the Head Tail property exists when there is a Strict, Containment, or Holding Intersection, and does not exist if a Middle Intersection is possible in the intersection model.

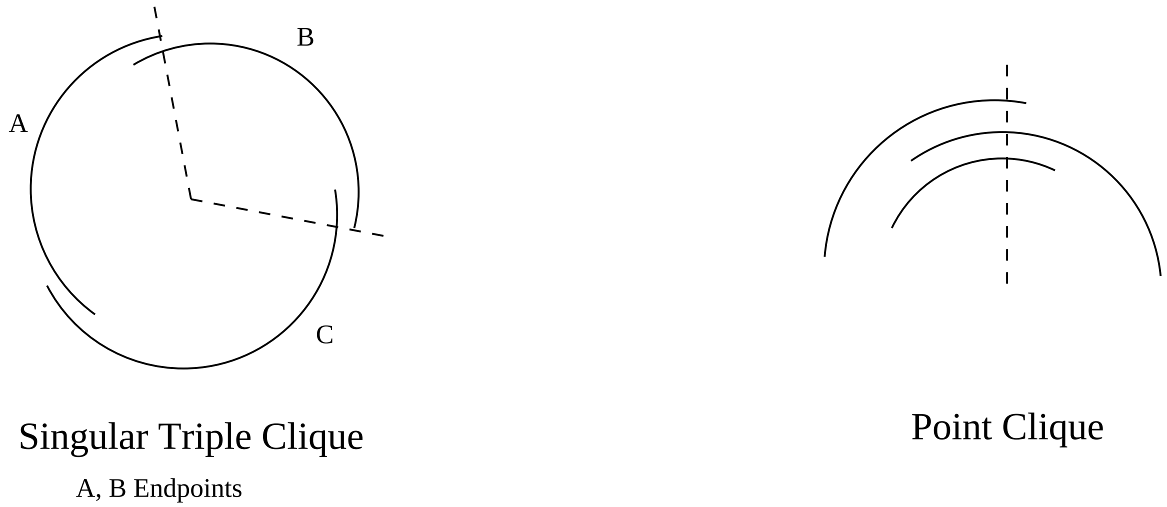

Figure 4: Types of Cliques

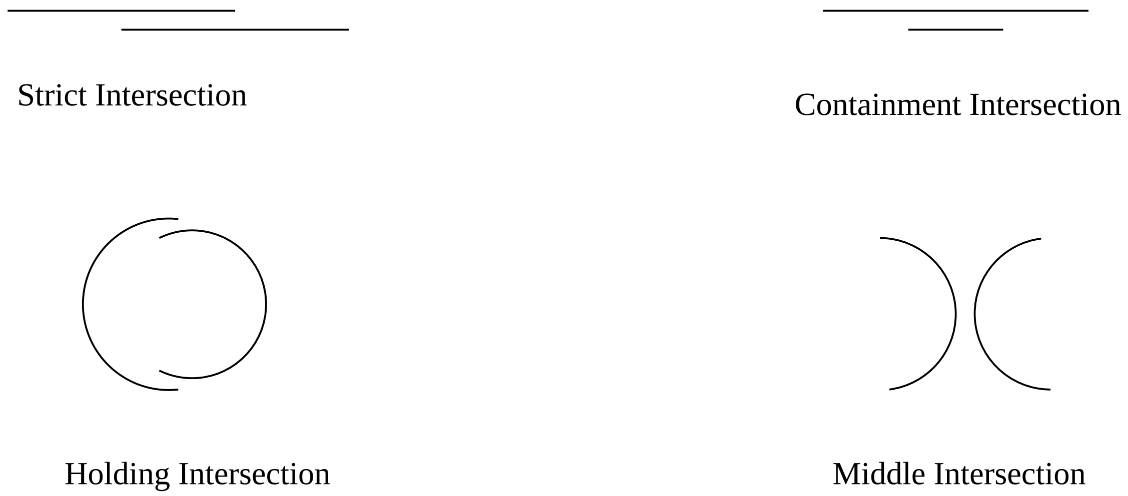

Figure 5: Types of Intersections

This property allows directions to be put on the edges in the intersection graph. As an arc that intersects another arc will contain the head or the tail of the other arc (or both), it is possible to set directions on the edges based on if the head of the intersecting arc is contained or the tail is contained. Assuming that the intersection model allows the arcs to never start or stop at the same place, the only problem arises when an arc contains the head and the tail of the intersecting arc. This may occur when the intersecting arc is strictly contained, in which case one arc contains the head and tail of the other and that arc does not contain either the head or the tail of the first arc. A problem also arises when the two arcs form a circle. The exact nature of the intersection will depend upon the intersection model being used. The fact that directions may be placed upon the edges means that an ordering may be created based on the intersection model.

If the intersection model allows the items to start and stop at the same point, a further problem is introduced in that if the only overlap is at that point, an arbitrary decision must be made as to which head or tail is in which item.

The fact that the Head Tail property is found in the 2-Circle graph makes the recognition algorithm easier to formulate in that it is known that the chordal arcs cannot intersect arcs on the left side of the circle in the intersection model. If the Head Tail property was not found in the 2-Circle graph, considerably more testing would be required, which would increase the time required for the completion of the algorithm.

The Head Tail property is found in the many of the previously studied intersection graphs. The interval graph class and the circular arc graph class also possess the Head Tail property, which is implied by the fact that these are subclasses of the 2-Circle graph class. We can find intersection graphs which do not possess the Head Tail property if we consider the family of intersection graphs of paths in a tree. If the tree is unrooted, the Head Tail property is not found, but in all other cases it is.

3 Recognition Algorithm

The recognition algorithm takes advantage of the fact that the 2- Circle graph is an extension to the Circular Arc graph class and so the $O(n^2)$ recognition algorithm developed for this class can be used[1]. It also uses the fact that two maximal cliques occur at the point that the chord intersects the circle in the intersection model. The only common vertices in these two cliques either pass through the chord or around at least half the circle. As it has been proven that there are a polynomial number of maximal cliques in a 2-Circle graph, even trying all pairs of these cliques is still polynomial. The recognition algorithm assumes that the cliques occur on the chord-circle intersection, and try to verify that the graph is of the 2-Circle class.

3.1 Recognition Algorithm for graph G

- If G is Circular Arc then G is a 2-Circle graph, else
- Find all maximal cliques, fail if greater than n^2 cliques
- For all pairs of maximal cliques with common vertices, C_1, C_2
  - V_1 = vertices in C_1 but not in C_2
  - V_2 = vertices in C_2 but not in C_1
  - V_C = vertices in C_1 and C_2
  - $G^1 = G - V_C$
  - if G^1 isn't Circular Arc then fail else
  - For each $v \in V_C$, if $G^1 \cup v$ is Circular Arc then remove v from V_C
  - Use the intersection model I^1 generated by the Circular Arc recognition algorithm for the following steps

- Find CH_1
  - Scan through arcs until all starting points of V_1 have been reached
  - Scan until next starting point of an arc, or the ending of an arc in V_1
  - Set CH_1 to that value
- Find CH_2 using V_2
- Find G_1
  - Clear $TEMP$
  - Starting at CH_1, scan until CH_2
  - If starting point of an arc is reached, place arc into $TEMP$
  - If ending point of an arc is reached and the arc is in $TEMP$, put the arc into G_1
- Find G_2, scanning from CH_2 to CH_1
- If $\exists v$ adjacent u where $v \in V_C, u \in G_1$
  then if $\exists v$ adjacent u where $v \in V_C, u \in G_2$ then fail
  else if $G - G_2$ is Circular Arc then G is a 2-Circle graph, else fail
  else if $G - G_1$ is Circular Arc then G is a 2-Circle graph, else fail

3.2 Proof and Discussion

The algorithm initially checks if the graph is Circular Arc. As the Circular Arc class is a subset of the 2-Circle class, this catches the easiest case.

Assuming this is not the case, the algorithm then finds the maximal cliques in the graph. It also uses this step to make sure that the graph can be a 2-Circle graph and that the algorithm remains polynomial by counting the maximal cliques found, as if to many maximal cliques are found the graph cannot be a 2-Circle graph.

The main portion of the algorithm consists of a loop which considers all neighbouring pairs of maximal cliques. The cliques can be assumed to be neighbours, that is they possess at least one vertex in common, because if they did not, no arcs would pass through the chord and the first step in the algorithm would have recognized the graph. The following steps all assume that the cliques chosen occur at the chord-circle intersection.

It is then checked that removing the chordal arcs will transform G into a Circular Arc graph, as would obviously be the case in a 2-Circle graph. After this, it is checked if any of the vertices common to both cliques are arcs on the circle, as opposed to being arcs passing through the chord. As any chordal arcs will connect adjacencies impossible in a Circular Arc graph, the Circular Arc recognition algorithm is used to verify the nature of the arc.

The intersection model generated by the Circular Arc recognition algorithm is then used to find the position of the chord. As each clique includes all arcs that pass over the chord-circle intersection, the position of the chord can be found by finding where all arcs unique to the clique overlap.

The arcs are then divided up into arcs solely to the left of the chord and arcs solely to the right of the chord. The fact that the collection of arcs on the left cannot intersect any of the chordal arcs, which is obvious from the intersection model, is used to ensure that the objects passing through the chord are arcs as opposed to trees. The fact is also used to find which is the left and right group of arcs. A final check is done to ensure that removing the arcs to the left of the chord creates a Circular Arc graph.

3.3 Time Required

This algorithm runs in polynomial time. This can be shown by looking at each step in the algorithm. Finding all maximal cliques requires $O(n^2)$ time as it has been previously shown that this is the maximum number of maximal cliques.

The next step in the algorithm checks all pairs of maximal cliques. As this step takes the square of the number of cliques, the time required for this step is $O(n^4)$.

The next step calls the Circular Arc recognition algorithm as many times as there are vertices in V_C. As this varies with n, and the time required for the recognition algorithm is $O(n^2)$, the time for this step is $O(n^3)$.

The following steps involve classifying the arcs found and the remaining vertices not assigned to an arc. As each vertex is looked at a constant number of times, this requires $O(n)$ time.

The total time required is therefore $O(n^2) + O(n^4)(O(n^3) + O(n))$, which comes out to $O(n^7)$.

3.4 Other Approaches and Improvements

The most obvious way to improve the recognition algorithm would be to narrow the number of cliques that are assumed to designate the position of the chord. One approach that I tried was to classify cliques in terms of how disconnected the graph became when the clique was removed from the graph. This approach was abandoned when the only unique feature of all possible chordal cliques found was that removing both cliques caused the graph to be disconnected. As this would require checking all pairs of cliques, it was not considered a useful trait.

Another improvement in the algorithm could come about by finding another method of deducing if a vertex common to both cliques passed through the chord or not. It seems likely that comparing the vertex to the intersection model already created at that time should allow such a classification to be made, but the exact method id not obvious.

Another approach considered looked at building up the graph by adding one or more vertices at a time while using the Circular Arc recognition algorithm and other rules to ensure the resulting graph was 2-Circle. This was found to be impractical as it was found that the order and combination in which the vertices were added to the graph could cause incorrect assumptions on the part of the algorithm.

Another method considered was to modfy the recognition algorithm of the circular arc class. However, this was found to be impractical as the recognition algorithm uses the relationship between the circular arc and bipartite graph classes, which does not exist for the 2-Circle class.

4 Maximum Weight Clique Algorithm

Given a known 2-Circle graph, an $O(mn)$ algorithm for the Maximum Weight Clique may be found by relying on the similarity to Circular Arc graphs, and using an O(mn) algorithm for Circular Arc graphs [3]. The algorithm requires that the endpoints of the arcs have been sorted and ranked in clockwise order, that the position of the chord is specified, and that the arcs going through the chord are so marked.

ALGORITHM

- V_C = arcs that pass through the chord
- $G_1 = G - V_C$
- C_1 is set to the maximum clique of G_1, using Circular Arc algorithm
- b = bottom chord position
- t = top chord position
- S = collection of arcs where both endpoints are after b and before t
- C_2 is set to the maximum clique of $G - S$, using the Circular Arc algorithm
- Max Clique = $max(C_1, C_2)$

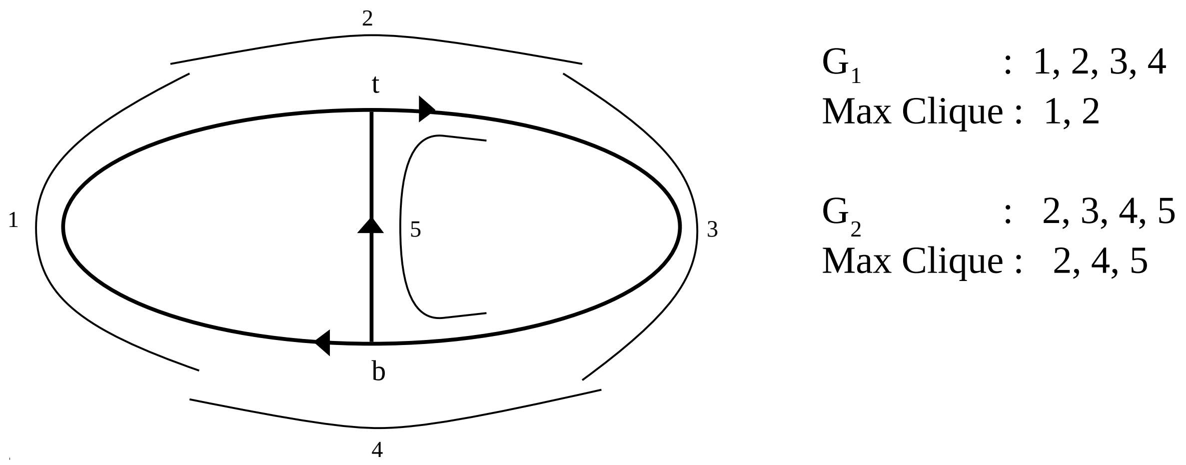

Figure 6: Maximum Clique Example

PROOF

G_1 is a circular arc graph. This is obvious.

If the maximum clique does not contain a chordal arc, then C_1 will be the maximum clique. This is obvious.

G_2 is a circular arc graph. Taking out the arcs that begin and end on the left of the chord creates a circular arc graph because all the arcs remaining on the left side of the circle intersect the arcs passing through the chords. This means that the intersection model may be drawn with the chordal arcs going all the way around the left side of the circle.

If the maximum clique contains a chordal arc, then C_2 will be the maximum clique. Because of the directions put on the underlying circle of the graph, no vertex in S can intersect a chordal arc and therefore cannot be in the maximum clique. Taking out S forces the graph into a circular graph and therefore the circular arc algorithm can be used.

TIME REQUIRED

The Circular Arc maximum clique algorithm is called twice, and all of the arcs must be looked at in order to classify them. This requires $2O(mn) + O(n)$ time. Therefore, the algorithm requires $O(mn)$ time.

5 Conclusion

The recognition algorithm for the 2-Circle graph is polynomial, but should not be assumed to be the most efficient algorithm possible. The algorithm relies heavily upon an efficient algorithm for the circular arc graph class. The maximum weight clique algorithm also takes advantage of the similarity of structure to the Circular Arc graph class in order to produce an efficient algorithm.

The Head Tail property presents a way of classifying different types of intersection graphs and may lead to a common method to recognize intersection graphs that possess this property. The Head Tail property also shows a relationship between the commonly known intersection graphs, and an idea as to their relative complexity.

References

[1] Eschen, E.M. and Spinrad, J.P. An $O(n^2)$ Algorithm for Circular-Arc Graph Recognition. In *ACM-SIAM Symposium on Discrete Algorithms (A Conference on Theoretical and Experimental Anaylsis of Discrete Algorithms)*, 1993.

[2] Gavril, F. The Intersection Graphs of Subtrees in Trees are Exactly the Chordal Graphs. *J. Combinatorial Theory 16*, 1 (Feb. 1974), 47-56.

[3] Hsu, W.L. Maximum Weight Clique Algorithm for Circular-Arc Graphs and Circle Graphs. *SIAM J. Computing 14*, 1 (Feb. 1985), 224-231.

[4] Korte, N. and Mohring, R.H. An Incremental Linear-time Algorithm for Recognizing Interval Graphs. *SIAM J. Computing 18*, 1 (Feb. 1989), 68-81.

[5] Monma, C. L. and Wei, V. K. Intersection Graphs of Paths in a Tree. *Journal of Combinatorial Theory 41*, 2 (Oct. 1986), 141-181.