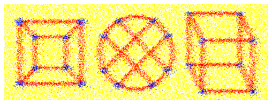
Journal of Graph Algorithms and Applications
http://jgaa.info/ vol. 22, no. 2, pp. 389–396 (2018)
DOI: 10.7155/jgaa.00476

A Faster Algorithm for Maximum Induced Matchings on Circle Graphs

Max Ward[1] *Andrew Gozzard*[1] *Michael Wise*[1] *Amitava Datta*[1]

[1]School of Computer Science and Software Engineering,
University of Western Australia, Perth

Abstract

Circle graphs have applications to RNA bioinformatics, computational chemistry, and VLSI design. Additionally, many problems that are intractable on general graphs are efficient for circle graphs. This has driven research into algorithms for circle graphs. One well known graph problem is to find a maximum induced matching. This is NP-Hard, even for bipartite graphs. No algorithm for this problem that works directly on circle graphs has been proposed. However, since circle graphs are included in interval filament graphs, algorithms for this class can be applied to circle graphs. Unfortunately, this entails a large computational cost of $O(|V|^6)$ time. We propose an algorithm that operates directly on circle graphs, and requires only $O(|V|^3)$ time.

Submitted:	Reviewed:	Revised:	Accepted:	Final:
April 2018	June 2018	June 2018	July 2018	July 2018
		Published: August 2018		
	Article type: Concise paper		Communicated by: G. Liotta	

E-mail addresses: max.ward-graham@research.uwa.edu.au (Max Ward) andrew.gozzard@uwa.edu.au (Andrew Gozzard) michael.wise@uwa.edu.au (Michael Wise) amitava.datta@uwa.edu.au (Amitava Datta)

1 Introduction

Circle graphs are intersection graphs resulting from a set of chords. A more rigorous definition is given in Section 2.1. Circle graphs are notable for two reasons. First, many problems that are typically hard on general graphs are tractable for circle graphs [1, 8, 10, 11, 13, 15, 17]. Second, circle graphs have practical applications. For instance, they are used to model problems in VLSI design. These problems include channel and switch-box routing [16]. In addition, circle graphs correspond to ribonucleic acid (RNA) secondary structures. As such, algorithms for circle graphs have been prominent in bioinformatics, sometimes without the authors' realizing that they use circle graphs. For example, Nussinov *et al.* [14] solved the maximum weight independent set problem on a circle graph as a method for predicting a likely RNA secondary structure. Another example comes from RNA structure visualization where Auber *et al.* [2] needed to approximate maximum induced bipartite subgraphs for circle graphs. One can also see that the conflict graphs used by Auber *et al.* are circle graphs. Similarly, Bonsma & Breuer [3] used circle graphs to model the counting of benzenoid hydrocarbons and fullerenes.

The Maximum Induced Matching (MIM) problem is to find a maximum matching in a simple undirected graph G, such that the vertices of no two edges in the matching share an edge in G. Put another way, we are to find a matching in G such that the induced subgraph is also a matching. The MIM problem is NP-Hard for both general graphs, and for bipartite graphs, but can be solved in polynomial time for chordal graphs [4], and some intersection graphs [5]. This problem has never been directly solved for circle graphs to our knowledge. The best algorithms for finding a MIM in circle graphs are for super classes of circle graphs, and follow from the findings of Cameron [5].

We describe a new algorithm for finding a MIM in circle graphs. This algorithm works directly on circle graphs, and requires only $O(|V|^3)$ time. This is faster than existing algorithms which operate on super classes of circle graphs and require $O(|V|^6)$ time.

1.1 Existing Algorithms

Existing algorithms capable of finding a MIM in circle graphs are implied by the work of Cameron [5]. Cameron points out three interesting properties about some intersection graph classes that make this possible. These can be found in Properties 1 to 3. To understand these, some definitions must be given. An independent set is a set of vertices in a graph such that no two vertices in the set share an edge. The line-graph $L(G)$ of a graph G is constructed by making a vertex corresponding to each edge in G where two vertices in $L(G)$ share an edge iff the corresponding edges in G share a vertex. Finally, the square G^2 of a graph G is constructed by adding non-existing edges between any pair of vertices in G that are connected by a path of length two.

Property 1. *Given any graph G, $[L(G)]^2$ is in the same class as G.*

Property 2. *Given any graph G, a MIM in G is a maximum independent set in $[L(G)]^2$.*

Property 3. *There exists a polynomial time algorithm for the maximum independent set problem.*

If Properties 1 to 3 hold for a graph class, then there is a polynomial time algorithm for the MIM problem on that graph class. Interestingly, Cameron found that only Property 1 is not true for circle graphs since $[L(G)]^2$ for a circle graph G may not be a circle graph [5]. In contrast, all three properties hold for interval-filament graphs. Interval-filament graphs are intersection graphs of a set of curves (filaments) with their endpoints describing intervals on a line such that the filaments of two disjoint intervals cannot intersect. Interval-filament graphs include circle graphs [9]. This implies a polynomial time solution for the MIM problem on circle graphs, as all circle graphs are also interval-filament graphs. More recently, a polynomial time maximum weight independent set algorithm was found for outerstring graphs by Keil *et al.* [12]. Since outerstring graphs include interval filament graphs, and thus circle graphs, this also implies a polynomial time algorithm for the MIM problem on circle graphs.

Since the squared line-graph of some circle graph $G = (V, E)$ is a graph $[L(G)]^2 = (V', E')$ in which $|V'| \in O(|V|^2)$, the complexity analysis of finding a maximum independent set in $[L(G)]^2$ is subtle. Both the algorithm of Gavril, which finds the maximum weight independent set on an interval-filament graph [9], and the algorithm of Keil *et al.*, which finds a maximum weight independent set in an outerstring graph [12], have time complexity $O(|V|^3)$. This implies that the best known complexity for finding a MIM for a circle graph is $O(|V|^6)$. The algorithm we shall present, which works directly on circle graphs, and not on the squared line-graph, requires only $O(|V|^3)$ time.

2 An Algorithm For Finding a Maximum Induced Matching in Circle Graphs

2.1 Circle Graph Models & Definitions

In this section, we explain fundamental terminology and concepts for circle graphs. Later, we shall build on these to describe our algorithms for the MIM problem. This section is an extension from our previous work on circle graphs [18].

A *circle graph* is described by a set of chords on a circle as in Figure 1. Each chord represents a vertex, and two intersecting chords share an edge in the circle graph. We shall make some simplifications to our model of circle without loss of generality. In particular, we first assume an *interval-arc* model of a circle graph.

The interval-arc model of a circle graph is a convenient representation in which the circle is cut at an arbitrary point. After doing this, we are left with a collection of arcs on a line segment as can be seen in Figure 1. The arcs represent

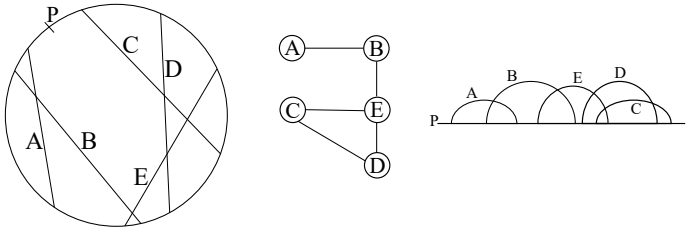

Figure 1: Three different representations of the same circle graph. The left panel shows the circle graph as a set of chords on a circle. The middle panel shows a corresponding graph representation. The right panel depicts an interval-arc model in which the circle has been cut at point P.

the chords on the original circle. Observe that the intersections between chords are preserved after cutting the circle.

Let us say that an interval-arc model contains N arcs ($|V| = N$), and that we assign each a unique number from 1 to N. Call these numbers *vertex numbers*, as they each correspond to a vertex in the circle graph. We can uniquely represent a circle graph as a permutation of these vertex numbers $\{1, 1, 2, 2, \ldots, N, N\}$. We will call this representation the *permutation model*. The left and right endpoints of an arc now correspond to the indexes of the first and last occurrence of that arc's number in the permutation model. The permutation can be constructed by enumerating the left-to-right order of endpoints in the interval-arc representation. In other words, the permutation model is an interval-arc model in which the extra space between endpoints is removed.

Given a vertex number x, we shall say $L(x)$ is the index of the first occurrence of x in the permutation model, and $R(x)$ is the index of the last occurrence. Also, we say that two different vertex numbers x and y *intersect* (and thus share an edge in the graph) iff $L(x) < L(y) < R(x) < R(y)$ or $L(y) < L(x) < R(y) < L(x)$.

In the permutation model, an induced matching is a set M of pairs of vertex numbers with the following properties:

Property 4. *For every $(x, y) \in M$, x intersects y.*

Property 5. *Neither x nor y intersect any other vertex number in M.*

A MIM is such a set that has the maximum possible cardinality.

2.2 Maximum Induced Matching Algorithm

We now describe a dynamic programming algorithm for finding a MIM in a circle graph. Call the permutation model of a circle graph p. Let us denote the vertex number at index i in p as p_i. We define $F(l, r)$ to be the cardinality of a MIM for the subrange of p from index l to r inclusive. Of course, $F(1, |p|)$ (assuming p is 1-indexed) is the cardinality of a MIM for our complete circle

graph. If l and r are not a valid range ($l > r$), then $F(l, r) = 0$. Otherwise, we can define the solution as follows:

$$F(l,r) = \max \begin{cases} F(l+1,r) \\ \max\{I_r(p_l,p_i) \ \forall \ i : l < i < R(p_l) \ \texttt{and} \ R(p_i) > R(p_l)\} \\ \qquad \texttt{if} \ L(p_l) = l \end{cases}$$
$$I_r(p_l,p_i) = F(L(p_l)+1, L(p_i)-1) + F(L(p_i)+1, R(p_l)-1) + F(R(p_l)+1, R(p_i)-1) + F(R(p_i)+1, r) + 1 \tag{1}$$

Let us unpack the logic behind Equation 1. First, observe that any MIM in a subrange must have a first vertex number. By first, we mean the vertex number in the MIM whose left endpoint is leftmost. The recurrence for $F(l, r)$ exploits this property. There are two cases to maximize over. The first considers $F(l+1, r)$, which covers the cases where the first vertex number in a MIM does not have its left endpoint at l. Conversely, in the second case, we cover all cases where the first vertex number in the MIM does have its leftmost endpoint at l, which is only possible if l is the index of a left endpoint ($L(p_l) = l$). Clearly, a MIM must either have its first vertex number at l, or not, so these cases are sufficient to define a solution.

A property of a MIM in a circle graph needs to be understood before explaining the correctness of the second case of $F(l, r)$. Consider any intersecting pair (p_l, p_i) of vertex numbers in an induced matching. We know that no other vertex number can intersect either. This means that every other vertex number in an induced matching must have both its endpoints fully contained between the endpoints of p_l and p_i, or completely to their left or right. Correspondingly, we define $I_r(p_l, p_i)$ to be the cardinality of a MIM given that p_l and p_i intersect, and that we are considering only the subrange from $L(p_l)$ to r inclusive. This needs only to sum up the cardinalities of MIMs for the subranges between the endpoints of p_l and p_i, and the remaining part of the subrange on the right. A visualization of this can be found in Figure 2.

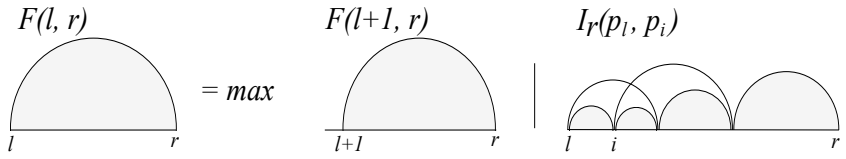

Figure 2: A representation of the two cases used to compute the recurrence in Equation 1. Unshaded arcs represent vertex numbers that are taken to be in a MIM. Shaded arcs represent solutions to $F(l, r)$, that is, a MIM for a subrange.

The function $I_r(p_l, p_i)$ is used in the second case of $F(l, r)$. Recall that we are assuming that l is the index of the left endpoint of the first vertex in a MIM

for the subrange $[l, r]$. If p_l is in a MIM, then it must intersect exactly one other vertex number p_i. Also, as p_l is first, $R(p_i) > R(p_l)$. This means that vertex numbers that occur between l and $R(p_l)$, and which have their right endpoints right of $R(p_l)$ are the only possible options for p_i. For each of these, the function I can be used to the find the corresponding MIM. Thus, all possibilities for the second case of $F(l, r)$ are maximized over.

After solving the recurrence for $F(l, r)$ using dynamic programming, the table of sub-solutions can be used to reconstruct a MIM. This is achieved by re-applying the function definition and recursively following the series of optimal sub-problems leading to the final solution. Alternatively, a table of the optimal choices can be maintained corresponding to each call to F, and the optimal solution can be traced by following these choices. Both of these techniques are classical approaches to recovering solutions to optimization problems after dynamic programming [6, 7].

2.3 Complexity Analysis

The size of the state space for F is $O(|V|^2)$. Observe that the number of indexes into a permutation model p is exactly $2|V|$, since every vertex number occurs exactly twice. The state in our dynamic programming recurrence comprises two indexes into p of which there are $O(|V|^2)$ possibilities. The final space complexity of our algorithm is also thus $O(|V|^2)$.

For each of the valid states for F, both cases described in Section 2.2 must be computed. The first takes $O(1)$ time, as it only considers a single option. The second takes $O(|V|)$ time, as it involves iteration through a range of indexes. This implies a final time complexity of $O(|V|^3)$.

3 Final Remarks

We have presented a faster algorithm to find a MIM in a circle graph. To our knowledge, this means that the best known time complexity for this problem is reduced from $O(|V|^6)$ to $O(|V|^3)$. However, we have not proved a lower bound, and wonder if a better complexity is possible. An interesting corollary to our algorithm is that it can be modified to work when endpoints are shared. This modified variant has time complexity $O(k^5)$ where k is the number of unique endpoints in the circle graph. This can be significant if many endpoints are shared [3, 18].

We wonder if our algorithm can be extended to more general graph classes. For example, we conjecture that our algorithm can be adapted to polygon-circle graphs and interval-filament graphs. Polygon-circle graphs generalize circle graphs as they are the intersection graphs of a set of polygons inscribed on a circle. Interval-filament graphs also generalize circle graphs and are defined in Section 1.1. To our knowledge, no algorithm faster than finding the maximum independent set in $[L(G)]^2$ is known, which implies an $O(|V|^6)$ algorithm for both.

References

[1] A. Apostolico, M. J. Atallah, and S. E. Hambrusch. New clique and independent set algorithms for circle graphs. *Discrete Applied Mathematics*, 36(1):1–24, 1992. doi:10.1016/0166-218X(92)90200-T.

[2] D. Auber, M. Delest, J.-P. Domenger, and S. Dulucq. Efficient drawing of rna secondary structure. *J. Graph Algorithms Appl.*, 10(2):329–351, 2006. doi:10.7155/jgaa.00131.

[3] P. Bonsma and F. Breuer. Counting hexagonal patches and independent sets in circle graphs. *Algorithmica*, 63(3):645–671, 2012. doi:10.1007/s00453-011-9561-y.

[4] K. Cameron. Induced matchings. *Discrete Applied Mathematics*, 24(1-3):97–102, 1989. doi:10.1016/0166-218X(92)90275-F.

[5] K. Cameron. Induced matchings in intersection graphs. *Discrete Mathematics*, 278(1-3):1–9, 2004. doi:10.1016/j.disc.2003.05.001.

[6] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms, Third Edition*. The MIT Press, 3rd edition, 2009.

[7] S. R. Eddy. What is dynamic programming? *Nature biotechnology*, 22(7):909, 2004. doi:10.1038/nbt0704-909.

[8] F. Gavril. Algorithms for a maximum clique and a maximum independent set of a circle graph. *Networks*, 3(3):261–273, 1973. doi:10.1002/net.3230030305.

[9] F. Gavril. Maximum weight independent sets and cliques in intersection graphs of filaments. *Information Processing Letters*, 73(5-6):181–188, 2000. doi:10.1016/S0020-0190(00)00025-9.

[10] F. Gavril. Minimum weight feedback vertex sets in circle graphs. *Information Processing Letters*, 107(1):1–6, 2008. doi:10.1016/j.ipl.2007.12.003.

[11] W.-L. Hsu. Maximum weight clique algorithms for circular-arc graphs and circle graphs. *SIAM Journal on Computing*, 14(1):224–231, 1985. doi:10.1137/0214018.

[12] J. M. Keil, J. S. Mitchell, D. Pradhan, and M. Vatshelle. An algorithm for the maximum weight independent set problem on outerstring graphs. *Computational Geometry*, 60:19–25, 2017. doi:10.1016/j.comgeo.2016.05.001.

[13] N. Nash and D. Gregg. An output sensitive algorithm for computing a maximum independent set of a circle graph. *Information Processing Letters*, 110(16):630–634, 2010. doi:10.1016/j.ipl.2010.05.016.

[14] R. Nussinov, G. Pieczenik, J. R. Griggs, and D. J. Kleitman. Algorithms for loop matchings. *SIAM Journal on Applied mathematics*, 35(1):68–82, 1978. doi:10.1137/0135006.

[15] D. Rotem and J. Urrutia. Finding maximum cliques in circle graphs. *Networks*, 11(3):269–278, 1981. doi:10.1002/net.3230110305.

[16] N. A. Sherwani. *Algorithms for VLSI physical design automation*. Springer Science & Business Media, 2012.

[17] A. Tiskin. Fast distance multiplication of unit-monge matrices. In *Proceedings of the twenty-first annual ACM-SIAM symposium on Discrete Algorithms*, pages 1287–1296. Society for Industrial and Applied Mathematics, 2010. doi:10.1007/s00453-013-9830-z.

[18] M. Ward, A. Gozzard, and A. Datta. A maximum weight clique algorithm for dense circle graphs with many shared endpoints. *Journal of Graph Algorithms and Applications*, 21(4):547–554, 2017. doi:10.7155/jgaa.00428.