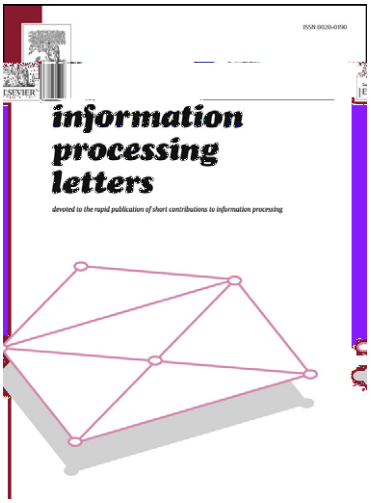


An output sensitive algorithm for computing a maximum independent set of a circle graph

Nicholas Nash, David Gregg

PII: S0020-0190(10)00139-0
DOI: 10.1016/j.ipl.2010.05.016
Reference: IPL 4318

To appear in: *Information Processing Letters*

Received date: 11 January 2010
Revised date: 19 May 2010
Accepted date: 20 May 2010

Please cite this article as: N. Nash, D. Gregg, An output sensitive algorithm for computing a maximum independent set of a circle graph, *Information Processing Letters* (2010), doi: 10.1016/j.ipl.2010.05.016

An Output Sensitive Algorithm for Computing a Maximum Independent Set of a Circle Graph

Nicholas Nash* David Gregg†

Abstract

We present an output sensitive algorithm for computing a maximum independent set of an unweighted circle graph. Our algorithm requires $O(n \min\{d, \alpha\})$ time at worst, for an n vertex circle graph where α is the independence number of the circle graph and d is its density. Previous algorithms for this problem required $\Theta(nd)$ time at worst.

1 Introduction

The *intersection graph* of a finite family of sets $X_1, \ldots, X_n$, is an undirected graph of n vertices $v_1, \ldots, v_n$ with an edge connecting v_i and v_j if and only if $X_i \cap X_j \neq \phi$. A *circle graph* is an undirected graph isomorphic to the intersection graph of a finite set of chords in a circle.

Finding a maximum independent set of a circle graph can be solved in time polynomial in the number of vertices of the graph, Gavril [4] presented a $\Theta(n^3)$ algorithm while others developed $\Theta(n^2)$ algorithms [3, 5, 9].

Apostolico *et al* [1, 2] solve the problem in $\Theta(nd)$ time and space, where d is a parameter known as the *density* of the circle graph. Valiente [11] solved the problem in $\Theta(nd)$ time and only $\Theta(n)$ space. Nash *et al* [8] experimentally studied the relative performance of optimized implementations of both of the preceding algorithms, showing that when suitably implemented, Valiente's algorithm performs better. As we note in Section 6 a variation of the algorithm described in this paper has been experimentally observed to significantly out-perform the best performing previous algorithm.

Other problems that are NP-complete for general graphs can also be solved in polynomial time for circle graphs, for example Tiskin [10] has shown an $O(n \log^2 d)$ time algorithm for the maximum clique problem. There are also problems that are NP-complete for both circle graphs and general graphs, such as minimum dominating set [7].

*Dept. of Computer Science, Trinity College Dublin, Ireland. Work supported by the Irish Research Council for Science, Engineering and Technology (IRCSET), nashn@cs.tcd.ie.

†Lero, Trinity College Dublin, dgregg@cs.tcd.ie.

The contribution of this paper is an algorithm for computing a maximum independent set of an unweighted circle graph in $O(n \min\{d, \alpha\})$ time, where α is the independence number of the circle graph.

2 Background

A circle graph can be represented either by chords in a circle or intervals on the real line, and the two representations may easily be obtained from one another [4]. Figure 1 shows an example of these representations together with the circle graph they give rise to. We can assume without loss of generality that no two chords (or intervals) share an end-point, since if two chords share an end-point we can slightly move the end-point of one of the chords without changing the circle graph [4].

An interval representation of an n vertex circle graph can be encoded as a permutation σ of $\{1, ..., 2n\}$ where the end-points of the intervals are formed by pairs $(\sigma_{2k-1}, \sigma_{2k})$, for $1 \leq k \leq n$. We refer to this as a σ-representation of the circle graph. Note that this makes use of the assumption that no intervals share an end-point. An arbitrary set of intervals can be transformed into this form in $O(n \log n)$ time.

In this paper we make use solely of this interval representation. The *density* of an interval representation is the maximum number of intervals crossing any point on the real line. Two intervals are said to *overlap* if neither contains the other and they are not disjoint. If two intervals overlap then their corresponding chords in the circle intersect. A maximum independent set is then a set of mutually non-overlapping intervals with cardinality at least as large as any other mutually non-overlapping set of intervals. Further background on circle graphs and related graph families is provided by Golumbic [6].

3 A Naive Algorithm

In this section we informally describe a naive algorithm for computing a maximum independent set of a circle graph, as well as introducing some preliminary notation and definitions. This naive algorithm forms the base for the output sensitive algorithm we introduce in Section 4.

We shall find it convenient to use the standard notation $[a, b] = \{x \in \mathbb{R} \mid a \leq x \leq b\}$ (for $a < b$) below and to speak of one interval containing another when this is true set-wise in $\mathbb{R}$. Given a set of intervals I we define $I_{q,m}$ to be the subset of those intervals which are contained in $[q, m]$. Throughout this paper, this set I should be regarded as having the form $\bigcup_{k=1}^{n}\{[\sigma_{2k-1}, \sigma_{2k}]\}$ for some given σ-representation of an n vertex circle graph. We denote the set of maximum independent sets of $I_{q,m}$ by $MIS_{q,m}$, and we denote their cardinality by $MIS[q, m]$. Finally, for an interval $i = [a, b]$ we define $CMIS_i = MIS_{a+1,b-1}$ and $CMIS[i] = MIS[a + 1, b - 1]$. Although MIS and $CMIS$ depend on the choice of a set of intervals I, we choose to suppress this dependence in the notation because it shall never lead to ambiguity. We now note some simple properties of $MIS[q, m]$.

Property 3.1 *If q is the right end-point of an interval, then $MIS[q,m] = MIS[q+1,m]$.*

This property follows since, if q is the right end-point of an interval, then $I_{q,m} = I_{q+1,m}$.

Property 3.2 *If q is the left end-point of an interval $i = [q,r]$ then*

$$MIS[q,m] = \begin{cases} MIS[q+1,m] \text{ if } r > m \\ \max\{MIS[q+1,m], 1 + CMIS[i] + MIS[r+1,m]\} \text{ otherwise} \end{cases}$$

Proof If $r > m$ then $I_{q,m} = I_{q+1,m}$ and so $MIS[q,m] = MIS[q+1,m]$. Assume $r \leq m$. Since $I_{q,m} \setminus \{i\} = I_{q+1,m}$ it follows that the cardinality of a maximum independent set of $I_{q,m} \setminus \{i\}$ is $MIS[q+1,m]$. Let V be a maximum cardinality independent set of $I_{q,m}$ that includes i. Clearly for any other interval $j \in I_{q,m}$ that does not overlap with i then either $j \in I_{q+1,r-1}$ or $j \in I_{r+1,m}$. Thus $|V| = 1 + CMIS[i] + MIS[r+1,m]$. Since when $r \leq m$ any $x \in MIS_{q,m}$ can either include or exclude i, it follows that $MIS[q,m] = \max\{MIS[q+1,m], 1 + CMIS[i] + MIS[r+1,m]\}$ in this case. ∎

Property 3.1 and 3.2 can be used to construct a simple dynamic programming algorithm for computing a maximum independent set of a circle graph. Figure 2 shows pseudo-code for this algorithm, which operates in $\Theta(n^2)$ time while using $\Theta(n)$ space. This algorithm evaluates $MIS[q,m]$ in an array $M[1 \ldots 2n]$ by increasing m from 1 to $2n$, and then evaluating the entries of the recurrence for this value of m by decreasing q from $m-1$ down to 1. When the algorithm terminates, $M[1] = MIS[1,2n]$. We note that a maximum independent set itself, and not just its weight, can easily be maintained while evaluating this recurrence.

The algorithm of Figure 2 is less efficient than Valiente's [11] $\Theta(nd)$ time algorithm, however, we include it because it motivates the output sensitive algorithm we introduce in the next section. The basic observation motivating the output sensitive algorithm is that, each time m is incremented in the naive algorithm, the number of cells in the array $M[1 \ldots 2n]$ whose value changes (in fact, increases) may be substantially smaller than the number of cells examined in the right-to-left scan of q from $m-1$ down to 1. Thus, if the set of cells that change can be determined efficiently, the resulting algorithm may be more efficient too. We formalize this intuition in the remainder of this paper.

4 An Output Sensitive Algorithm

Definition 4.1 *Given the σ-representation of an n vertex circle graph, the update set at m, $1 \leq m \leq 2n$, is*

$$S_m = \{q \mid MIS[q,m] > MIS[q,m-1] \text{ for } 1 \leq q \leq m\}$$

Proposition 4.2 *If $q \in S_m$ then $MIS[q,m] = 1 + MIS[q,m-1]$.*

Proof By assumption $MIS[q,m] = k + MIS[q,m-1]$, $k \geq 1$, and thus there must be an interval $i = [l,m]$ included in all $x \in MIS_{q,m}$ but not in any $y \in MIS_{q,m-1}$. For all $x \in MIS_{q,m}$ clearly $x \setminus \{i\}$ is an independent set of of $I_{q,m-1}$ and if $k > 1$ then its cardinality would exceed $MIS[q,m-1]$, which is a contradiction and thus $k = 1$. ∎

Definition 4.3 *(*UPDATE *algorithm) Given the σ-representation of an n vertex circle graph, an integer $1 \leq m \leq 2n$ and an array $M[1 \ldots 2n]$ such that $M[q] = MIS[q,m-1]$ for $1 \leq q \leq m-1$, and $M[q] = 0$ otherwise, and an array $C[1 \ldots n]$ such that for each interval $i = [l,r]$ with $r \leq m$ $C[i] = CMIS[i]$ (here $C[i]$ should be regarded as being indexed by the rank of i in the σ-representation of the circle graph) then we define the* UPDATE *algorithm as follows:*

If m is the left end-point of an interval, the algorithm terminates. Otherwise, there is an interval $i = [l,m]$ and the algorithm performs the assignment $M[l] \leftarrow 1 + C[i]$, and l is pushed onto a stack T.

The following procedure is then iterated until the stack T contains no entries. The top of the stack is popped into x. If $x > 1$ and $M[x] > M[x-1]$ then the update $M[x-1] \leftarrow M[x]$ is performed (we refer to this as a Type-1 update) and $x-1$ is pushed onto T. If there is an interval $j = [p, x-1]$ and $1 + C[j] + M[x] > M[p]$ then the assignment $M[p] \leftarrow 1 + C[j] + M[x]$ is performed (we refer to this as a Type-2 update) and p is pushed onto T, and the process iterates.

Pseudo-code for this algorithm is provided in Figure 3.

Lemma 4.4 *If any cell $M[q]$, $1 \leq q \leq m$ is modified by the* UPDATE *algorithm, then $M[q] = MIS[q,m]$ when the algorithm terminates.*

Proof Assume the algorithm iterates at least once until the stack T is empty, possibly modifying cells of M. If, for all indices x on the stack T we have $M[x] = MIS[x,m]$ we say T is *valid*. Assume that T is valid at the beginning of each loop iteration where any cell of M is modified. Assume that value of $M[p]$ has been modified by the algorithm on some iteration.

If p is a right end-point, then it must have been modified by a Type-1 update via an assignment of the form $M[p] \leftarrow M[p+1]$, where $p+1$ was popped from T and since T is valid $M[p] = MIS[p+1,m]$. Moreover, by Property 3.1 $MIS[p,m] = MIS[p+1,m]$, and thus $M[p] = MIS[p,m]$, as required. The algorithm then pushes p onto T, and so T is still valid after p is pushed onto it.

Otherwise, if $M[p]$ was modified and p is a left end-point of an interval $j = [p, x-1]$, then $M[p]$ may have been modified by either a Type-1 or a Type-2 update (but not both, see Proposition 4.2). If $M[p]$ was modified only by a Type-1 update, then $M[p] = M[p+1]$ and $M[p+1] \geq 1 + C[j] + M[x]$. If $M[p]$ was modified by a Type-2 update, then $M[p] = 1 + C[j] + M[x]$ and $1 + C[j] + M[x] \geq M[p+1]$. Thus, in the

case of either update, $M[p] = \max\{M[p+1], 1+C[j]+M[x]\}$. Since T is valid $M[p+1] = MIS[p+1, m]$ and $M[x] = MIS[x, m]$. Thus $M[p] = \max\{MIS[p+1, m], 1+CMIS[j]+MIS[x, m]\}$, and so by Property 3.2 $M[p] = MIS[p, m]$ as required. The algorithm then pushes p onto T, and so T is still valid after p is pushed onto it.

It follows immediately from the above that if T is valid at the beginning of some iteration of the loop, then T is valid at the beginning of the next iteration of the loop, and thus on every iteration of the loop. To complete the proof, we show T is valid before the first iteration of the loop.

If m is the right end-point of an interval $i = [l, m]$, then the first index pushed onto the stack is l. Preceding this, the algorithm performs the assignment $M[l] \leftarrow 1 + C[i]$. Note that $MIS[m+1, m] = 0$, and that $MIS[l+1, m] = MIS[l+1, m-1] = C[i]$, and thus by Property 3.2, $M[l] = MIS[l, m]$ following the assignment. As a result T is valid after l is pushed onto it. ∎

Definition 4.5 *If at some point during the execution of the* UPDATE *algorithm we have* $M[q] = MIS[q, m]$ *for all* $q \geq p$*, we say that* M *is fully updated at* p*.*

Property 4.6 *If* $M[p]$ *is not modified by the* UPDATE *algorithm and* M *is fully updated at* $p+1$ *then* $MIS[p, m] = MIS[p, m-1]$*.*

Proof If p is a right end-point and $M[p+1]$ was not modified by the algorithm then, since M is fully updated, $MIS[p+1, m-1] = MIS[p+1, m]$ but by Property 3.1 $MIS[p, m-1] = MIS[p+1, m-1]$ and $MIS[p, m] = MIS[p+1, m]$ and hence $MIS[p, m] = MIS[p, m-1]$, as required. If p is a right end-point and $M[p+1]$ was modified by the algorithm, then since no Type-1 update of $M[p]$ occured, we have $M[p] \geq M[p+1]$, implying $MIS[p, m-1] \geq MIS[p+1, m]$, but by Property 3.1 $MIS[p, m] = MIS[p+1, m]$, and hence $MIS[p, m-1] \geq MIS[p, m]$ but clearly $MIS[p, m-1] \leq MIS[p, m]$ since $I_{p,m-1} \subseteq I_{p,m}$, and hence $MIS[p, m] = MIS[p, m-1]$, as required.

Otherwise, if p is a left end-point of an interval $j = [p, x-1]$ and neither $M[p+1]$ nor $M[x]$ has been updated then $MIS[p+1, m] = MIS[p+1, m-1]$ and $M[x] = MIS[x, m-1]$, thus, by Property 3.2 $MIS[p, m] = \max\{MIS[p+1, m-1], 1+CMIS[j]+MIS[x, m-1]\}$ showing that at $MIS[p, m] = MIS[p, m-1]$ as required. Otherwise, if both of $M[p+1]$ and $M[x]$ have been updated then we have $M[p] \geq \max\{MIS[p+1, m], 1+CMIS[j]+MIS[x, m]\}$, since M is fully updated and neither a Type-1 or a Type-2 update occured. That is, $MIS[p, m-1] \geq MIS[p, m]$, implying as above that $MIS[p, m] = MIS[p, m-1]$, as is required. Lastly, we consider the cases where exactly one of $M[p+1]$ and $M[x]$ has been updated. By Property 3.2 $MIS[p, m-1] = \max\{MIS[p+1, m-1], 1+CMIS[j]+MIS[x, m-1]\}$. Assume that only $M[p+1]$ was updated, and since M is fully updated at $p+1$, $MIS[p+1, m] > MIS[p+1, m-1]$. But since $M[p]$ was not updated, and in particular there was no Type-1 update,

we have $MIS[p, m-1] \geq MIS[p+1, m]$, implying that $MIS[p, m-1] > MIS[p+1, m-1]$ and in turn that $MIS[p, m-1] = 1 + CMIS[j] + MIS[x, m-1]$, and once more since $M[p]$ was not updated $1 + CMIS[j] + MIS[x, m-1] \geq MIS[p+1, m]$, and hence $MIS[p, m] = 1 + CMIS[j] + MIS[x, m-1]$, and $MIS[p, m] = MIS[p, m-1]$, as required. The other case, where only $M[x]$ is updated is analogous. ∎

Lemma 4.7 *When the* UPDATE *algorithm terminates* $M[q] = MIS[q, m]$ *for* $1 \leq q \leq m$.

Proof If m is the left end-point of some interval then the algorithm can terminate without modifying M because $I_{q,m} = I_{q,m-1}$ for $1 \leq q \leq m$. Assume the algorithm iterates $z \geq 1$ times. On the k^{th} iteration of the algorithm, we denote the contents of the stack T at the beginning of that iteration as T_k. Moreover we denote the largest index on some T_k as $F(T_k)$. Note that the algorithm modifies only the cells $M[F(T_1)], \ldots, M[F(T_z)]$. Note also that for each $1 \leq k < z$ the algorithm modifies none of the cells $M[F(T_{k+1}) + 1], \ldots, M[F(T_k) - 1]$.

Consider any iteration, $1 \leq k < z$, and assume M is fully updated at $F(T_k)$. It follows by $F(T_k) - F(T_{k+1}) - 1$ applications of Property 4.6 that M is fully updated at $F(T_{k+1}) + 1$. Moreover, when the algorithm modifies $M[F(T_{k+1})]$, by Lemma 4.4, M will be fully updated at $F(T_{k+1})$. Thus if M is fully updated at $F(T_k)$ at the beginning of k^{th} iteration of the algorithm, it is fully updated at $F(T_{k+1})$ on the $(k+1)^{th}$ iteration. Note that at the beginning of the first iteration of the algorithm, M is fully updated at $F(T_1) = l$ noting that $I_{r,m} = I_{r,m-1}$ for $l < r \leq m$ and that $M[l] = MIS[l, m]$ by the same argument given at the end of in Lemma 4.4.

Finally, after the final, z^{th} iteration, a further $F(T_z) - 1$ applications of Property 4.6 show that M is fully updated at 1 when the algorithm terminates. ∎

Lemma 4.8 *The* UPDATE *algorithm terminates in* $O(1+|S_m|)$ *time and uses* $O(1+|S_m|)$ *space in addition to the space used by its input.*

Proof The algorithm only pushes an index q onto the stack T if $q \in S_m$. Moreover, by Proposition 4.2 such an index q can be pushed onto T at most once by the algorithm. Thus T can contain at most $|S_m|$ indices and since the algorithm uses at most constant space in addition to T the total space is $O(1+|S_m|)$ space (allowing for $S_m = \phi$).

Since the main loop of the algorithm iterates until T contains no entries, this loop can iterate at most $|S_m|$ times, with each iteration incurring constant time. Thus, the total time is $O(1 + |S_m|)$. ∎

Lemma 4.9 *Given an* n *vertex circle graph with independence number* α *then* $\sum_{q=1}^{2n} |S_q| \leq 2n\alpha$.

Proof Let $f(q)$ be the number of the update sets $\{S_1, \ldots, S_{2n}\}$ that q is an element of. Clearly $\sum_{q=1}^{2n} f(q) = \sum_{q=1}^{2n} |S_q|$. Also note that $f(1) \geq f(2) \geq \cdots \geq f(2n)$, and in particular that $\alpha = f(1)$. It follows that $\sum_{q=1}^{2n} f(q) \leq 2n\alpha$ and thus that $\sum_{q=1}^{2n} |S_q| \leq 2n\alpha$. ∎

Lemma 4.10 *Given the σ-representation of a circle graph, the cardinality of its maximum independent sets can be computed in $O(n\alpha)$ time and $O(n)$ space*

Proof We refer to the algorithm in Figure 4, here, UPDATE(M, C, m) denotes an invocation of the UPDATE algorithm of Definition 4.3, Figure 3 and Lemma 4.7.

On the m^{th} iteration of the loop, given $M[q] = MIS[q, m-1]$ for $1 \leq q < m$ and $C[i] = CMIS[i]$ for all interval $i \in I_{q,m-1}$ with $r < m$, by Lemma 4.7, UPDATE will modify M such that $M[q] = MIS[q, m]$ for $1 \leq q \leq m$. Examining Figure 3, C will be modified giving $C[i] = CMIS[i]$ for all $i \in I_{1,m}$.

Thus, after the m^{th} iteration of the loop, the inputs C and M required by UPDATE in the next iteration are available. Since on the first iteration the inputs required by UPDATE are trivially available, it follows that when the for loop terminates $M[1]$ holds the cardinality of the maximum independent sets of the circle graph.

For the time complexity, note that there are $2n$ calls to the UPDATE algorithm, requiring by Lemma 4.8, in total time $\sum_{m=1}^{2n} O(1 + |S_m|)$ which is $O(n\alpha)$ by Lemma 4.9.

For the space complexity, note that in addition to the $O(n)$ space required by the input, the only space used $O(n)$ for the array C and, by Lemma 4.8, at most $O(n)$ for the stack T. ∎

5 A Combined Algorithm

Having established the $O(n\alpha)$ time and $O(n)$ space algorithm of Lemma 4.10, we can combine it with Valiente's [11] $O(nd)$ time and $O(n)$ space algorithm giving a $O(n \min\{d, \alpha\})$ time algorithm. The following is the main result of this paper.

Theorem 5.1 *Given the σ-representation of a circle graph, the cardinality of its maximum independent sets can be computed in $O(n \min\{d, \alpha\})$ time and $O(n)$ space.*

Proof In $O(n)$ time and $O(1)$ space the parameter d can be computed by a simple iteration over the σ-representation. Now, the UPDATE algorithm of Figure 3 is modified such that if an update to a cell $M[q]$ gives $M[q] > d$, then the UPDATE algorithm simply sets a flag F and terminates.

To compute the cardinality of the maximum independent sets, the algorithm of Figure 4 is used with this modified UPDATE algorithm. When the algorithm terminates, if the flag F is not set, then it must be the case that $\alpha < d$ and hence the cardinality of the maximum independent sets has been determined in $O(n\alpha)$ time using at most $O(n)$ space, by Lemma 4.10.

On the other hand, if the flag F is set, it must be the case that $d < \alpha$. In this case, at most $O(nd)$ time has been incurred using $O(n)$ space. To complete the computation, Valiente's algorithm [11] is invoked on the σ-representation, requiring an additional $O(nd)$ time and $O(n)$ space.

Thus, the modified algorithm requires $O(n \min\{d, \alpha\})$ time. ∎

We note that a maximum independent set itself, and not just its cardinality, can easily be maintained by this algorithm within the same time and space bounds. Referring to Figure 4, a cell $C[i]$ of $C[1 \ldots 2n]$ can be thought of as representing an element of $CMIS_i$ rather than $CMIS[i]$, and $C[i]$ can then be represented by a linked list of all the intervals directly contained in an element of $CMIS_i$ (an interval j is *directly* contained in $x \in CMIS_i$ if $j \in x$ and there is no $k \in x$ such that j is contained in the element of $CMIS_k$ represented by $C[k]$). The same linked list representation can be used in place of $M[1 \ldots 2n]$. This simple representation of a maximum independent set is illustrated by Valiente [11].

6 Conclusion

This paper has described a $O(n \min\{d, \alpha\})$ time and $O(n)$ space algorithm for computing a maximum independent set of an unweighted circle graph. Future work could attempt to extend coverage to weighted circle graphs. In addition, we note that in experiments we have observed an algorithm based on the output sensitive approach described in this paper to perform excellently in practice. In these experiments, a variation of the algorithm described in this paper was used. This variation appears to have favourable constant factors and operates in time $O(n \min\{d, \alpha \log n\})$, it was compared to an optimized implementation of the previous best algorithm, whose experimental performance was documented in [8]. The output sensitive algorithm was observed to offer much improved performance (by a factor of between 3 and 7). Moreover, in these experiments we have observed a class of circle graphs with $d = \Theta(n)$ and $\alpha = \Theta(\sqrt{n})$. We leave a full description of these experimental results to future work.

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.

[2] A. Apostolico, M. J. Atallah, and S. E. Hambrusch. Erratum: New clique and independent set algorithms for circle graphs (discrete applied mathematics 36 (1992) 1—24). *Discrete Applied Mathematics,* 41(2):179—180, 1993.

[3] T. Asano, H. Imai, and A. Mukaiyama. Finding a maximum weight independent set of a circle graph. *IEICE Transactions*, E74(4):681—683, 1991.

[4] F. Gavril. Algorithms for a maximum clique and a maximum independent set of a circle graph. *Networks*, 3(3):261—273, 1973.

[5] O. Goldschmidt and A. Takvorian. An efficient algorithm for finding a maximum weight independent set of a circle graph. *IEICE Transactions*, E77-A(10):1672—1674, 1994.

[6] M. C. Golumbic. Algorithmic Graph Theory and Perfect Graphs *(Annals of Discrete Mathematics, Vol 57)*. North-Holland Publishing Co., Amsterdam, The Netherlands, The Netherlands, 2004.

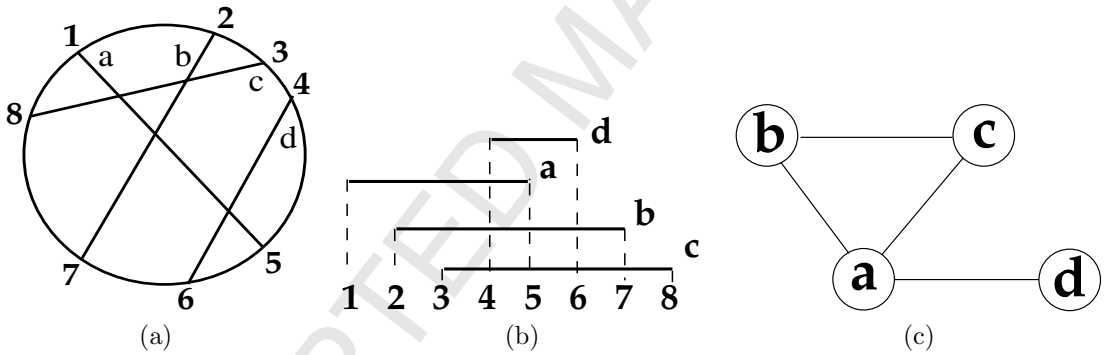
[7] J. M. Keil. The complexity of domination problems in circle graphs. *Discrete Applied Mathematics,* 42(1):51—63, 1993.

[8] N. Nash, S. Lelait, and D. Gregg. Efficiently implementing maximum independent set algorithms on circle graphs. *J. Exp. Algorithmics*, 13(1.9):1—34, 2009.

[9] K. J. Supowit. Finding a maximum planar subset of a set of nets in a channel. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 6(1):93—94, 1987.

[10] A. Tiskin. Semi-local string comparison: algorithmic techniques and applications. *CoRR*, abs/0707.3619, 2009.

[11] G. Valiente. A new simple algorithm for the maximum-weight independent set problem on circle graphs. In T. Ibaraki, N. Katoh, and H. Ono, editors, *ISAAC, volume 2906 of Lecture Notes in Computer Science*, pages 129—137. Springer, 2003



(a) (b) (c)

Figure 1: The chords in the circle of (a) give rise to the circle graph in (c). The intervals in (b) also give rise to the circle graph of (c). The interval representation in (b) has density 4.

Input: A σ-representation of an n vertex unweighted circle graph.
Output: The cardinality of the circle graph's maximum independent sets.
$M[1 \ldots 2n] \leftarrow 0$
$C[1 \ldots n] \leftarrow 0$
for $m \leftarrow 1$ **to** $2n$ **do**
    if m is the right end-point of an interval $i = [l, m]$ **then**
        $C[i] \leftarrow M[l + 1]$
    endif
    for $q \leftarrow m - 1$ **downto** 1 **do**
        $M[q] \leftarrow M[q + 1]$
        if q is the left end-point of an interval $j = [q, r]$ and $r \leq m$ **then**
            $M[q] \leftarrow \max\{M[q + 1], 1 + C[j] + M[r + 1]\}$
        endif
    endfor
endfor

Figure 2: Pseudo-code for a simple $\Theta(n^2)$ time and $\Theta(n)$ space algorithm for computing the cardinality of a circle graph's maximum independent sets.

Input: The σ-representation of an n vertex circle graph together with an integer m and the arrays M and C as defined in Definition 4.3.
Output: M such that $M[q] = MIS[q, m]$ for $1 \leq q \leq m$.
if m is the right end-point of an interval $i = [l, m]$ **then**
    $M[l] \leftarrow 1 + C[i]$
    PUSH(T, l)
    while TOP$(T) \neq$ NIL
        $x \leftarrow$ POP(T)
        if $x > 1$ AND $M[x] > M[x - 1]$ **then**
            $M[x - 1] \leftarrow M[x]$
            PUSH$(T, x - 1)$
        endif
        if there is an interval $j = [p, x - 1]$ AND
            $1 + C[j] + M[x] > M[p]$ **then**
            $M[p] \leftarrow 1 + C[j] + M[x]$
            PUSH(T, p)
        endif
    endwhile
endif

Figure 3: Pseudo-code for the UPDATE algorithm of Definition 4.3.

Input: A σ-representation of an n vertex unweighted circle graph.
Output: The cardinality of the circle graph's maximum independent sets.
$M[1 \ldots 2n] \leftarrow 0$
$C[1 \ldots n] \leftarrow 0$
for $m \leftarrow 1$ **to** $2n$ **do**
    if m is the right end-point of an interval $i = [l, m]$ **then**
        $C[i] \leftarrow M[l + 1]$
    endif
    UPDATE(M, C, m)
endfor

Figure 4: Pseudo-code for a $O(n\alpha)$ time and $O(n)$ space algorithm for computing the cardinality of a circle graph's maximum independent sets. UPDATE(M, C, m) denotes an invocation of the UPDATE algorithm of Definition 4.3, Figure 3 and Lemma 4.7.