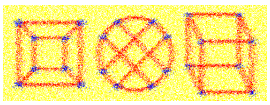

Journal of Graph Algorithms and Applications
http://jgaa.info/ vol. 30, no. 1, pp. 323–338 (2026)
DOI: 10.7155/jgaa.v30i1.3137

A New Simple Algorithm for Computing Maximum Weight Induced Forests in Circle Graphs

Nicholas Nash

Submitted: Nov. 2025 Accepted: August 2026 Published: August 2026

Article type: Regular Paper Communicated by: Md. Saidur Rahman

Abstract. This paper describes a new, simple algorithm for computing a maximum weight induced forest in a circle graph. The algorithm requires $O(n^3m)$ time and $O(n^3)$ space. In the unweighted case, an algorithm operating in $O(n^2mk)$ time and $O(nk^2)$ space is described, where k is the cardinality of a maximum induced forest in the circle graph. The previously described algorithms require $O(n^7)$ time and $O(n^3)$ space. The new algorithm is simple to describe and implement within the stated bounds. We also note the existence of an (impractical) $O(n^{4.686})$ time algorithm in the unweighted case.

1 Introduction

A circle graph is an undirected (finite, simple) graph that is isomorphic to the intersection graph of some finite set of chords drawn in a circle [17]. Figure 1 gives an example graph. Circle graphs have found applications in VLSI design [7, 28], RNA secondary structure and folding [6, 24], container stowage [2], register allocation [11] and computational chemistry [3]. Further background on circle graphs and related graph classes is given by Golumbic [18] and Spinrad [27].

Several problems that are NP-Complete for general graphs can be solved in polynomial time for circle graphs. For example, for an n vertex circle graph, a maximum clique can be found in $O(n \log^2 d)$ time [29] and a maximum independent set can be found in $O(n \min\{d, \alpha\})$ time [23]. Here $d = O(n)$ is parameter of the circle graph's representation known as its density, while α denotes the cardinality of a maximum independent set.

Other problems, such as Hamiltonian cycle, minimum dominating set and computing the chromatic number of a circle graph are known to be NP-Complete [9, 21, 30]. Gavril [14, 15] showed that a maximum induced forest of a circle graph can also be computed in polynomial time, by describing an algorithm requiring $O(n^7)$ time and $O(n^3)$ space.

The contribution of this paper is a simple algorithm requiring $O(n^3m)$ time and $O(n^3)$ space to compute a maximum weight induced forest of an n vertex circle graph having m edges. For an unweighted circle graph having maximum induced forests of cardinality k, an $O(n^2mk)$ time algorithm is also noted, requiring $O(nk^2)$ space.

E-mail address: ncnash@pm.me (Nicholas Nash)

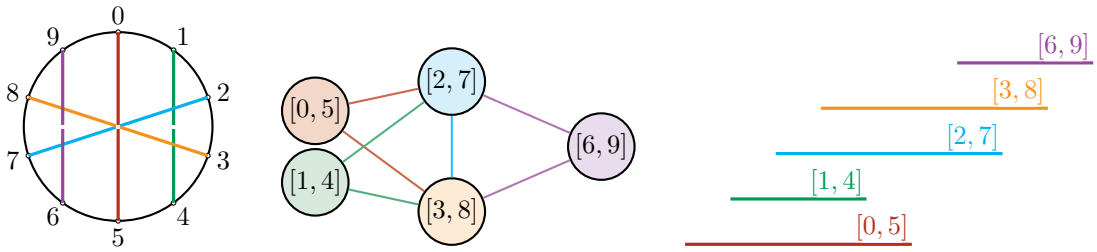


Figure 1: From left-to-right, a chord model of a circle graph, the circle graph itself, and an interval model of the circle graph. Each chord of the chord model corresponds to a vertex of the circle graph. Similarly each interval corresponds to a vertex of the circle graph. There is an edge between two vertices of the circle graph if and only if the corresponding two chords in the chord model intersect. Moreover, two chords of the chord model intersect if and only if the two corresponding intervals in the interval model intersect without one containing the other.

The existing algorithm of Gavril [14] for computing a maximum weight induced forest of a circle graph operates by computing a maximum weight independent set of a suitably defined interval-filament graph [13], the algorithm in this paper operates by constructing explicit recurrences for the problem directly on an interval model of the circle graph. This is arguably simpler, allows re-use of several key maximisations and leads to a more efficient algorithm.

2 Preliminaries

An arbitrary graph of n vertices and m edges can be recognised as a circle graph in $O(n+m)$ time [25]. The first polynomial time circle graph recognition algorithms ranged in time from $O(n^7)$ to $O(n^3)$ time [4, 12, 22]. These algorithms operate in terms of the *split decomposition* [8] of the input graph. Improvements to this technique lead to an $O(n^2)$ recognition algorithm [26], followed much later by the first subquadratic algorithm [16], and finally the recent linear time algorithm [25]. The output of these recognition algorithms is a representation of the circle graph as a set of chords, or, equivalently, as a set of intervals on the real line. Figure 1 illustrates. Note that such so-called chord models or interval models of a circle graph are not necessarily unique, see [16, 25].

In this paper, we assume that a set of n intervals $I(G)$ with distinct end-points chosen from integers in $[0, 2n-1]$ is given. We refer to $I(G)$ as an *interval model* of G. The corresponding circle graph, G, has a vertex for each interval in $I(G)$, and an edge between two vertices when the corresponding intervals intersect, but neither contains the other. We say that any two such intervals *overlap*.

Given a node, u, in a circle graph G, we denote by $i(u) = [\ell_u, r_u]$, the corresponding interval in $I(G)$, and its given arbitrary real numbered weight as $w(u)$. We write $i(u) \subset i(v)$ when interval $i(u)$ is contained set-wise in $i(v)$, and we say $i(u)$ is contained in $i(v)$. If either $i(u) \subset i(v)$ or $i(v) \subset i(u)$, we say u and v are *nested*.

An induced forest F in G is an acyclic induced subgraph of G. Given two nodes u, v in such an F, we say that v is a right-child of u if $i(v)$ contains only r_u and not ℓ_u. Similarly, we say v is a left-child of u in F if $i(v)$ contains only ℓ_u and not r_u. Nodes in the forest may have multiple children, as Figure 2(a) illustrates.

In addition, we refer to the set of nodes connected to u in F via a right-child as the right descendants of u, and denote them by $R(u)$, and similarly we refer to left descendants, and denote

them $L(u)$. For a node u, we denote all its descendants by $D(u)$, that is, $D(u) = L(u) \cup R(u)$.

A maximum induced forest in an undirected graph G is the largest set of vertices that can be chosen in G such that the induced subgraph is acyclic, and it is equivalent to finding the minimum feedback vertex set of G, that is, the smallest set of vertices in G whose deletion yields an acyclic graph. The latter problem is NP-Complete in general [20]. Gavril [14] described an $O(n^7)$ time and $O(n^3)$ space dynamic programming algorithm for computing a maximum induced forest of a circle graph.

We next note several properties of circle graphs that our algorithm relies upon, re-using the notion of an *oriented* induced forest from [14]. We provide proofs below in order to give a self-contained account.

Without loss of generality, the algorithm produces an *oriented* maximum induced forest:

Definition 1. *An induced forest of a circle graph G is **oriented**, if and only if, the root of each tree in the forest corresponds to the interval in $I(G)$ with smallest left end-point in that tree.*

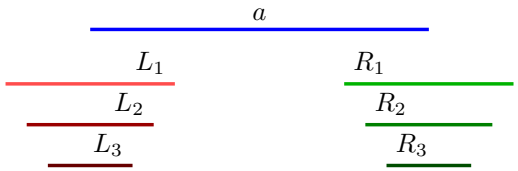
Perhaps the simplest consequence of this definition is the following:

Lemma 1. *The root of any tree in an oriented forest has no left descendants.*

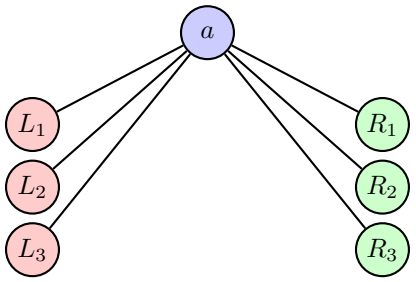
Proof: Suppose that the root r has a left descendant. Then the first node on the path from r to that descendant is a left child v of r, and so $\ell_v < \ell_r$. This contradicts the choice of r as the node with smallest left end-point in its tree. □

Lemma 2. *For any node in an induced forest of a circle graph, all the intervals corresponding to the left children of that node are nested, and similarly for the right children. Figure 2 illustrates.*

Proof: Consider an induced forest F of a circle graph. Let a be a node in F with two right children c_1 and c_2. There cannot be an edge between c_1 and c_2, since F is acyclic. So $i(c_1)$ and $i(c_2)$ must either be disjoint, or one must contain the other. But $i(c_1)$ and $i(c_2)$ both contain r_a and so they cannot be disjoint. Thus they are nested. The same argument applies to two left children. □



(a) An interval model of a circle graph, with a distinguished as root of an induced forest, noting that intervals corresponding to left and right children must be nested within each other.



(b) A maximum induced forest for the interval model of (a), with a as root with left children L_1, L_2, L_3 and right children R_1, R_2, R_3.

Figure 2: The nested property of children, noted in Lemma 2.

We shall also use the following property of oriented induced forests, which is Lemma 1 of Gavril [14]. We include a proof in our terminology.

Fact 1. *Let w be a node of an oriented induced forest of a circle graph, and let $d \in D(w)$. Then $i(w) \not\subset i(d)$.*

Proof: Suppose, for a contradiction, that $i(w) \subset i(d)$. Let r be the root of the tree containing w, and write the path from w to r as $w = x_0, x_1, \ldots, x_k = r$. We show by induction that $i(x_j) \subset i(d)$ for every j.

The claim is true for $j = 0$ by assumption. Suppose that $i(x_j) \subset i(d)$ for some $j < k$. Since x_j and x_{j+1} are adjacent, their intervals overlap. Therefore $i(d)$ intersects $i(x_{j+1})$. The intervals $i(d)$ and $i(x_{j+1})$ cannot overlap, since the resulting edge from d to x_{j+1}, together with the tree path between these nodes, would create a cycle. Moreover, $i(d)$ cannot be contained in $i(x_{j+1})$, since then $i(x_j) \subset i(d) \subset i(x_{j+1})$ contradicting the fact that $i(x_j)$ and $i(x_{j+1})$ overlap. It follows that $i(x_{j+1}) \subset i(d)$.

Thus $i(r) \subset i(d)$. In particular, $\ell_d < \ell_r$, contradicting the fact that the root of an oriented tree has the smallest left end-point in that tree. □

The following lemma is useful in decomposing the problem of computing a maximum induced forest of a circle graph:

Lemma 3. *Let x and y be nodes in an induced forest of a circle graph, neither the ancestor of the other, having lowest common ancestor w. If $i(x) \subset i(y)$, then every node on the path from x to w, excluding w, has its corresponding interval contained in $i(y)$.*

Proof: Write the path from x to w as $x_0, x_1, \ldots, x_k$, i.e. $x = x_0$ and $w = x_k$. We show by induction that $i(x_j) \subset i(y)$ for every $j < k$. The base case $j = 0$ is true by assumption. Now suppose $j < k - 1$ and $i(x_j) \subset i(y)$. Since x_j is adjacent to x_{j+1} in the tree, the intervals $i(x_j)$ and $i(x_{j+1})$ overlap. Hence $i(y)$ intersects $i(x_{j+1})$. It cannot overlap $i(x_{j+1})$, because the edge from y to x_{j+1} would imply the existence of two distinct paths from w to x_{j+1}, one including y, and one excluding y. That is, a cycle. Nor can $i(y) \subset i(x_{j+1})$, since then $i(x_j) \subset i(y) \subset i(x_{j+1})$, contradicting the fact that $i(x_j)$ and $i(x_{j+1})$ overlap. Therefore $i(y)$ must contain $i(x_{j+1})$. □

Again as simple consequences of the definitions of interval overlap and acyclicity, we derive a condition that will allow recursing on "inner" sub-trees in the algorithms of Sections 3 and 4:

Lemma 4. *Let z be a node of an oriented induced forest of a circle graph, with two right children u and v such that $i(u) \subset i(v)$. Define*

$$p = \max\Big(\{\ell_v\} \cup \{r_x : x \in L(v)\}\Big), \qquad q = \min\Big(\{r_v\} \cup \{\ell_y : y \in R(v)\}\Big).$$

Then every $d \in \{u\} \cup D(u)$ satisfies $i(d) \subset [p + 1, q - 1]$. The same statement holds when u and v are both left children.

Proof: We prove the right-child case. We first note that every node in $D(v)$ has its interval disjoint from $i(u)$. Let $t \in D(v)$. The intervals $i(t)$ and $i(u)$ cannot overlap, since the resulting edge from t to u, together with the tree path between these nodes, would create a cycle. If $i(t) \subset i(u)$, then Lemma 3 would imply $i(v) \subset i(u)$, contradicting $i(u) \subset i(v)$. Finally, suppose that $i(u) \subset i(t)$. Since $i(u)$ overlaps $i(z)$, the interval $i(t)$ intersects $i(z)$. It cannot overlap $i(z)$ without creating a cycle; it cannot be contained in $i(z)$, since this would imply $i(u) \subset i(z)$; and it cannot contain $i(z)$, by Fact 1. Thus $i(t)$ and $i(u)$ are disjoint.

Now let $x \in L(v)$, and let x_1 be the first node after v on the path from v to x. Since x_1 is a left child of v then $\ell_{x_1} < \ell_v < \ell_u$. Since $i(x_1)$ is disjoint from $i(u)$, it lies to the left of $i(u)$. Every

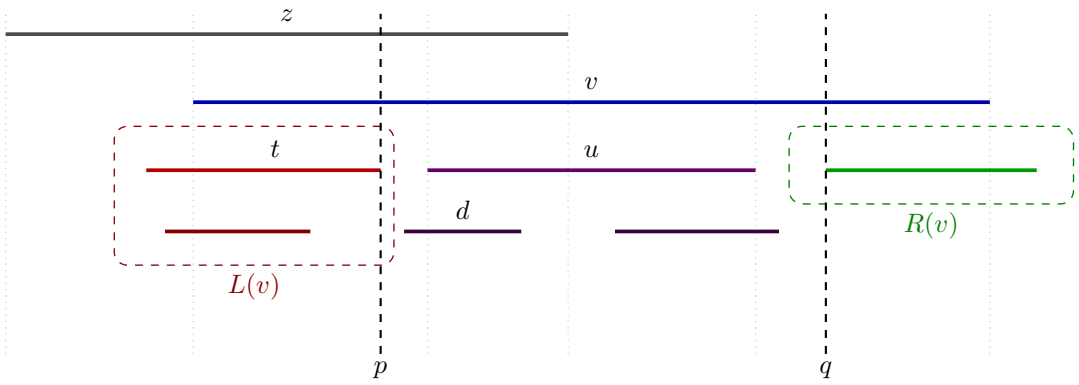


Figure 3: An illustration of Lemma 4. In the forest corresponding to these intervals, z has two right children u and v, with $i(u) \subset i(v)$. All the descendants of u are contained on either side by the descendants of v (or by v itself if it doesn't have descendants on one side or the other).

subsequent interval on the path is also disjoint from $i(u)$ and overlaps the preceding interval, so the path cannot pass to the right of $i(u)$. Hence $r_x < \ell_u$ for every $x \in L(v)$. Symmetrically, $\ell_y > r_u$ for every $y \in R(v)$. Together with $i(u) \subset i(v)$, this gives $i(u) \subset [p+1, q-1]$.

We next show that neither p nor q belongs to any interval $i(d)$ with $d \in \{u\} \cup D(u)$. Let x be the node whose end-point gives p, so either $x = v$ and $p = \ell_v$, or $x \in L(v)$ and $p = r_x$. Suppose that $p \in i(d)$. The nodes x and d lie in the distinct branches below v and u, respectively, so their intervals cannot overlap. Since p is an end-point of $i(x)$ and all end-points are distinct, $i(d)$ cannot be contained in $i(x)$. Therefore $i(x) \subset i(d)$. If $x = v$, then $i(v) \subset i(d)$ directly. Otherwise, Lemma 3 implies $i(v) \subset i(d)$. Hence $i(u) \subset i(v) \subset i(d)$. This is impossible when $d = u$, and when $d \in D(u)$ it contradicts Fact 1. Therefore $p \notin i(d)$. The same argument shows that $q \notin i(d)$.

We finish by induction on the distance from u. The base case follows from $i(u) \subset [p+1, q-1]$ established above. Suppose that $i(s) \subset [p+1, q-1]$ for some $s \in \{u\} \cup D(u)$, and let d be a child of s. If d is a left child, then $\ell_d < \ell_s < r_d < r_s < q$. Since $p \notin i(d)$ and $p < \ell_s < r_d$, we must have $\ell_d > p$. If d is a right child, then $p < \ell_s < \ell_d < r_s < r_d$. Since $q \notin i(d)$ and $\ell_d < r_s < q$, we must have $r_d < q$.

In either case, $i(d) \subset [p+1, q-1]$ and the result follows by induction. □

The following lemma, which Figure 4 illustrates, allows decomposing the problem of finding an oriented maximum induced forest with a given root in a circle graph recursively into left and right sub-problems:

Lemma 5. *Let u be a node of an oriented induced forest of a circle graph. Then for any $a \in L(u)$ and $b \in R(u)$, the intervals $i(a)$ and $i(b)$ are disjoint.*

Proof: We first observe that no right descendant of u can contain ℓ_u. A right child of u does not contain ℓ_u by definition. Now let t be a right descendant of u that is not a child of u, and suppose for contradiction that $\ell_u \in i(t)$. Then $i(t)$ intersects $i(u)$. The two intervals cannot overlap, since the resulting edge from u to t, together with the tree path from u to t, would create a cycle. Moreover, $i(t)$ cannot be contained in $i(u)$, since it contains the end-point ℓ_u. Thus $i(u) \subset i(t)$, contradicting Fact 1. Therefore no right descendant of u contains ℓ_u. Symmetrically, no left descendant of u contains r_u.

Now let c be the child of u on the path from u to a, and let d be the child of u on the path from u to b. Then c is a left child of u, while d is a right child of u.

The intervals $i(a)$ and $i(b)$ cannot overlap, since the resulting edge from a to b, together with the tree path from a to b through u, would create a cycle. Hence, if they are not disjoint, one must contain the other.

Suppose first that $i(a) \subset i(b)$. By Lemma 3, every node on the path from a to u, excluding u, has its interval contained in $i(b)$. In particular $i(c) \subset i(b)$. Since c is a left child of u, its interval contains ℓ_u. Consequently $\ell_u \in i(b)$, contradicting the fact proved above that no right descendant of u contains ℓ_u.

The case $i(b) \subset i(a)$ is symmetric. □

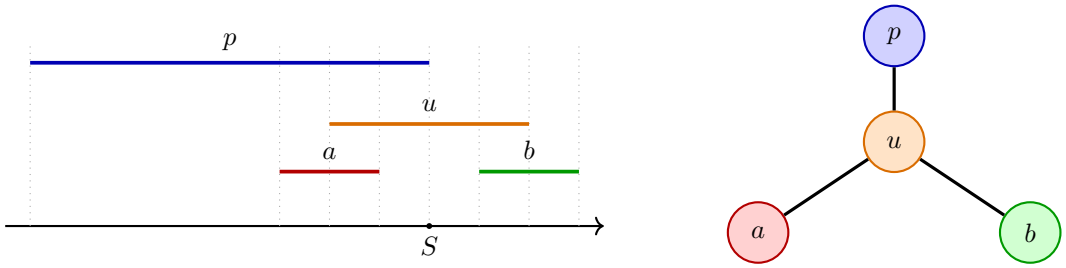


Figure 4: An example used to give the gist of Lemma 5. The intervals a, b must be disjoint, and so must their descendants.

We next describe a very simple algorithm that makes use of these properties to compute a maximum induced forest of a circle graph in $O(n^6)$ time and $O(n^3)$ space.

3 A New (Naive) Algorithm

Using the lemmas of Section 2 we now derive recurrences that give rise to a new, simple algorithm for computing the maximum induced forest of a circle graph. Naively evaluating the recurrences requires $O(n^6)$ time and $O(n^3)$ space, this is already a slight improvement over Gavril's algorithm [14].

We begin by noting simple recurrences that can be used for computing a maximum induced *tree* of a circle graph.

Definition 2. *Given a circle graph $G = (V, E)$ and an interval model $I(G)$, let $RT[u, L, R]$, for $\ell_u < L \leq r_u \leq R$, be the weight of a maximum induced tree in G, rooted at $u \in V$, but not including $w(u)$ in its weight, using only right descendants of u whose corresponding intervals are within $[L, R]$.*

Similarly we define $LT[u, L, R]$ for $L \leq \ell_u \leq R < r_u$. Next we introduce notation for so-called eligible left and eligible right children:

Definition 3. *Given a circle graph $G = (V, E)$, and an interval model $I(G)$, then for any interval $i(u)$ in an induced forest of G, we denote the eligible right children of u by $RC(u, L, R)$, that is, the set of nodes whose intervals have both end-points within $[L, R]$, and also contain r_u but not ℓ_u. We similarly define $LC(u, L, R)$ as those nodes whose intervals are in $[L, R]$ that contain only ℓ_u but not r_u.*

We now show how to compute LT and RT in Lemma 6. Figure 5 illustrates.

Lemma 6.

$$RT[u,L,R] = \max_{v\in RC(u,L,R)} \max_{\substack{p\in[\ell_v,r_u-1]\\ q\in[r_u+1,r_v]}} \Big(\underbrace{w(v)}_{\textit{include } v} + \underbrace{LT[v,L,p]}_{\textit{left subtree of } v} + \underbrace{RT[v,q,R]}_{\textit{right subtree of } v} + \underbrace{RT[u,p+1,q-1]}_{\textit{inner subtree of } u}\Big). \quad (1)$$

$$LT[u,L,R] = \max_{v\in LC(u,L,R)} \max_{\substack{p\in[\ell_v,\ell_u-1]\\ q\in[\ell_u+1,r_v]}} \Big(\underbrace{w(v)}_{\textit{include } v} + \underbrace{LT[v,L,p]}_{\textit{left subtree of } v} + \underbrace{RT[v,q,R]}_{\textit{right subtree of } v} + \underbrace{LT[u,p+1,q-1]}_{\textit{inner subtree of } u}\Big). \quad (2)$$

with the convention that the value is 0 *if no feasible* v,p,q *exist.*

Proof: Consider an oriented (not necessarily maximum) induced tree, T, rooted at node u, where u and all intervals corresponding to its descendants have both end-points within $[L,R]$. Since T is oriented, by Lemma 1 u may only have right children. Assume u has a right child v, then the subtree rooted at v is comprised of nodes $P \subseteq L(v)$ and $Q \subseteq R(v)$. Let the corresponding sets of intervals be P' and Q'. By Lemma 5, for any $a \in P'$, $b \in Q'$, then $a \cap b = \phi$. Thus, if p is the largest right end-point in P' (or ℓ_v if $L(v) = \phi$), and q the smallest left end-point in Q' (or r_v if $R(v) = \phi$), then all intervals in P' have both end-points within $[L,p]$, and all intervals in Q' have both end-points within $[q,R]$. And so the maximum weight of the subtree rooted at and including v is $w(v)+LT[v,L,p]+RT[v,q,R]$. By Lemmas 2 and 4, all other children of u have intervals with both end-points within $[p+1,q-1]$, and the maximum weight of this subtree is $RT[u,p+1,q-1]$. Thus the maximum weight of the induced tree T, with root u, having chosen v as a right-child is $w(v)+LT[v,L,p]+RT[v,q,R]+RT[u,p+1,q-1]$. Moreover the subproblems rooted at v in $[L,p]$ and $[q,R]$, together with the subproblem rooted at u in $[p+1,q-1]$ combine to give an induced tree. It follows that $RT[u,L,R]$ is given by maximising over all possible choices of v,p,q □

Naturally, $LT[u,L,R]$ has an analogous computation in terms of left children.

To compute a maximum induced forest in G, as opposed to a maximum induced tree, we now define the following, in exact analogy with Definition 2. First we introduce the notion of a *right-forest*:

Definition 4. *Let F be an oriented induced forest containing a node u, and let T_u be the component of F containing u. We call F a* ***right-forest rooted at*** *u if u is the root of T_u, and every other component of F either lies strictly to the right of r_u, or is nested in some node of T_u other than u. A component is nested in a node v if every interval corresponding to a node of that component is strictly contained in $i(v)$.*

Definition 5. *Given a circle graph $G=(V,E)$ and an interval model $I(G)$, define $RF[u,L,R]$, for $u \in V$ and $\ell_u < L \le r_u \le R$, as the maximum total weight of a right-forest rooted at u, excluding the weight of u, in which every node other than u has its corresponding interval contained in $[L,R]$.*

Similarly we define $LF[u,L,R]$ for $L \le \ell_u \le R < r_u$. Next we define the weight of an oriented maximum induced forest without reference to a root:

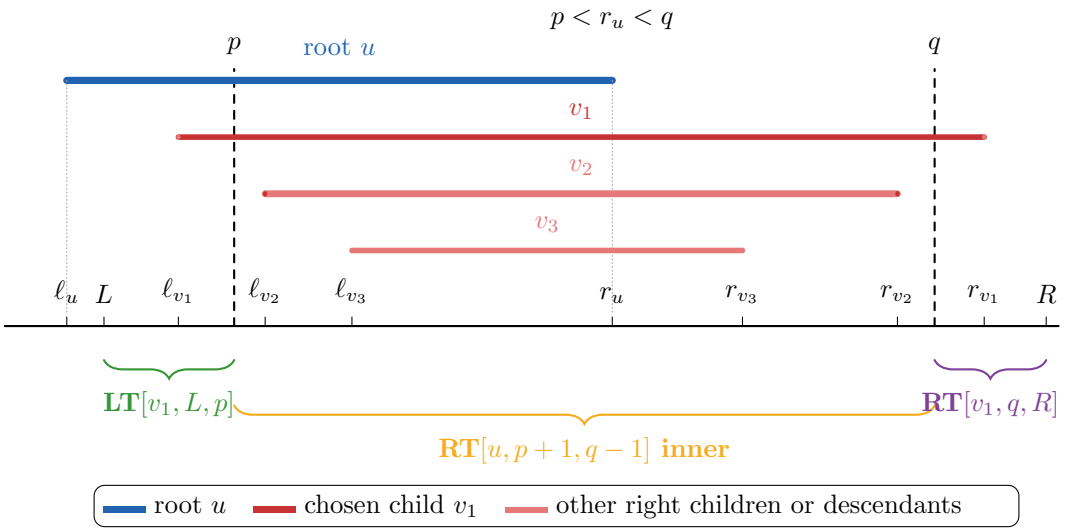


Figure 5: An illustration of how Recurrence 1 requires choosing a right child of some interval u, solving the left-tree problem *LT within* u, solving the inner-tree problem within that *child*, and solving another right-tree problem to the right of the child. For example, after attaching v_1 as a child to u in the maximum induced tree, it may be that then v_2 (and in turn v_3) is attached as a result of solving the inner subproblem.

Definition 6. *Given a circle graph G and an interval model $I(G)$, define $B[a, R]$ as the weight of an oriented maximum induced forest in G with all corresponding intervals having both end-points within $[a, R]$.*

The weight of a maximum induced forest in the corresponding circle graph is then $B[0, 2n-1]$. The following simple Lemma shows how a recurrence can be used to compute B

Lemma 7.

$$B[a, R] = \max_{u : i(u) \subseteq [a,R]} \; \max_{s \in [\ell_u, r_u - 1]} \Big(w(u) \; + \; B[\ell_u + 1, \; s] \; + \; RF[u, \; s+1, \; R] \Big) \tag{3}$$

Proof: Consider an oriented (not necessarily maximum) induced forest F, with all intervals in $[a, R]$, with node u chosen as the root of the tree with smallest left end-point. Assume the right descendants of u are all within $[s+1, R]$ for some $s \in [\ell_u, r_u - 1]$. The largest weight of the induced forest including these right descendants of u is $RF[u, s+1, R]$. Since u has smallest left end-point, it has no left descendants, and the maximum weight induced forest of all other nodes (corresponding to intervals in $[\ell_u + 1, s]$) is simply $B[\ell_u + 1, s]$. Thus we can compute $B[a, R]$ simply by maximising over choices of u, s. □

To now compute $RF[u, L, R]$, a recurrence exactly analogous to Recurrence 1 is used.

Lemma 8.

$$RF[u,L,R] = \max\Bigg\{ B[r_u+1,R], \max_{v\in RC(u,L,R)} \max_{\substack{p\in[\ell_v,r_u-1]\\ q\in[r_u+1,r_v]}} \Big(w(v) + LF[v,L,p] + RF[v,q,R] + RF[u,p+1,q-1]\Big)\Bigg\}. \quad (4)$$

Proof: Consider an optimal solution to the subproblem $RF[u,L,R]$. If no right child of u is chosen, then nothing remains to be placed in the interval window $[L,r_u]$. The remaining solution is therefore simply an oriented induced forest whose intervals lie in $[r_u+1,R]$, and its maximum weight is $B[r_u+1,R]$.

Now suppose that u does have at least one right child. By Lemma 2, the right children of u are nested, so there is a unique outermost right child v, namely the one whose interval contains the intervals of all the other right children of u. Let $P = L(v)$ and $Q = R(v)$ inside the chosen solution, and define

$$p = \max\Big(\{\ell_v\} \cup \{r_x : x \in P\}\Big), \qquad q = \min\Big(\{r_v\} \cup \{\ell_y : y \in Q\}\Big).$$

By Lemma 5, every interval belonging to P is disjoint from every interval belonging to Q. Hence all vertices of P lie in $[L,p]$, and all vertices of Q lie in $[q,R]$. Moreover, by Lemma 4, every node belonging to the subtree of any other right child of u lies in $[p+1,q-1]$.

Thus the solution splits into four parts:

$$\underbrace{w(v)}_{\text{choose } v} + \underbrace{LF[v,L,p]}_{\text{everything on the left of } v} + \underbrace{RF[v,q,R]}_{\text{everything on the right of } v} + \underbrace{RF[u,p+1,q-1]}_{\text{the remaining inner part of } u}.$$

Moreover the subproblems combine to produce an induced forest, and taking the maximum over these possibilities gives the recurrence. □

LF is again exactly analogous. For completeness we give the definition:

$$LF[u,L,R] = \max\Bigg\{ B[l_u+1,R], \max_{v\in LC(u,L,R)} \max_{\substack{p\in[\ell_v,\ell_u-1]\\ q\in[\ell_u+1,r_v]}} \Big(w(v) + LF[v,L,p] + RF[v,q,R] + LF[u,p+1,q-1]\Big)\Bigg\}. \quad (5)$$

These recurrences can be directly evaluated in $O(n^6)$ time and $O(n^3)$ space: There are $O(n^3)$ entries and the maximisation over v,p,q requires $O(n^3)$ time per cell. The next section notes that it is very simple to improve this to $O(n^3m)$ time also in $O(n^3)$ space.

4 An $O(n^3m)$ Time Algorithm

An $O(n^3m)$ time algorithm can be obtained simply by re-using the inner maxima in Recurrences 3, 4 and 5. The following facts are merely substitutions of inner terms within these recurrences. That is we define:

$$G[u,R] = \max_{s\in[\ell_u, r_u-1]} \big(w(u) + B[\ell_u+1, s] + RF[u, s+1, R]\big) \tag{6}$$

And so we can re-write Recurrence 3 as:

$$B[a,R] = \max_{u:i(u)\subseteq[a,R]} G[u,R] \tag{7}$$

Thus, all values of G, and B can be computed in $O(n^3)$ time. Similarly, we can define, for a node u having right child v, and $l_v \le p < r_u \le R$

$$MR[u,v,p,R] = \max_{q\in[r_u+1, r_v]} \Big(RF[v,q,R] + RF[u,p+1,q-1]\Big) \tag{8}$$

We re-write Recurrence 4 as:

$$RF[u,L,R] = \max\left\{ B[r_u+1,R], \max_{v\in RC(u,L,R)} \max_{p\in[\ell_v, r_u-1]} \Big(w(v)+LF[v,L,p]+MR[u,v,p,R]\Big)\right\} \tag{9}$$

Both MR and RF can be evaluated in $O(n^5)$ time. MR need only be stored for a single parent, u, and child v, pair at a time and so it requires only $O(n^2)$ space in total. Similarly, LF can be evaluated in $O(n^5)$ time. Algorithm 1 shows a concrete evaluation order of these recurrences to compute a maximum induced forest of a circle graph. We show the evaluation of MR and RF in Algorithms 3 and 2, we omit the symmetrical evaluation of ML and LF. With reference to Algorithm 1 we obtain:

Theorem 1. *A maximum weight induced forest of an n vertex circle graph $G = (V, E)$, $m = |E|$ can be computed in $O(n^3 m)$ time and $O(n^3)$ space.*

Proof: Fix a value of R in Algorithm 1. For a fixed node u with $r_u \le R$, Algorithm 2 computes $MR[u,v,p,R]$ for all $v \in RC(u,0,R)$ and all feasible p. For each such v, the loop over q has length $O(n)$ and the loop over p has length $O(n)$, so the total time spent by Algorithm 2 for this u is $O(n^2|RC(u,0,R)|)$.

Similarly, for fixed u and R, Algorithm 3 computes all values $RF[u,L,R]$. For each $v \in RC(u,0,R)$, the loop over p has length $O(n)$ and the loop over L has length $O(n)$, and so Algorithm 3 also requires $O(n^2|RC(u,0,R)|)$ time for this u.

Summing over all nodes u with $r_u \le R$, the total time spent by Algorithms 2 and 3 for this fixed R is $O(n^2 \sum_{u\in V} |RC(u,0,R)|)$. Now for any edge in the circle graph between vertices u and v, at most one of the following conditions holds:

- $u \in RC(v,0,R)$ and $v \notin RC(u,0,R)$, or
- $v \in RC(u,0,R)$ and $u \notin RC(v,0,R)$

Therefore $\sum_{u\in V} |RC(u,0,R)|$ counts every edge in the circle graph at most once. That is, $\sum_{u\in V} |RC(u,0,R)| \le m$ and so the total time spent by Algorithms 2 and 3 for this fixed R is $O(n^2 m)$.

By symmetry, the same bound holds for the left-sided computations performed by `Compute-ML` and `Compute-LF`. Finally, Algorithm 4 spends only $O(n)$ time for each u, and hence $O(n^2)$ time for each fixed R. Thus, for each fixed R, the total time is $O(n^2 m)$. There are $O(n)$ possible values of R, and therefore the total running time of Algorithm 1 is $O(n^3 m)$.

For the space bound, the tables RF and LF each have $O(n^3)$ entries, while B has $O(n^2)$ entries. The temporary tables MR and ML are computed for one fixed pair u, R at a time and each uses only $O(n^2)$ space. Hence the total space usage is $O(n^3)$. □

In the unweighted case, we also note that a maximum cardinality induced forest of a circle graph may be computed in $O(n^2mk)$ time, where k is its cardinality. This is because the terms in the inner-most maxima over s, q and p in Recurrences 6, 8 and 9 respectively, are monotone sequences of at most k distinct values. A simple implicit representation of the sequences involved immediately allows computing these maxima in $O(k)$ time with $O(k^2)$ space per vertex, thus $O(nk^2)$ space in total is required in the unweighted case. In the next section, we sketch a more efficient algorithm for the unweighted case.

5 A Subquintic Time Algorithm

When $m = \Theta(n^2)$, the algorithms of Section 4 require $O(n^5)$ time. For this dense case, we now briefly note an asymptotically more efficient $o(n^5)$ time algorithm for finding an unweighted maximum induced forest in a circle graph, and more generally when all vertex weights are nonnegative integers bounded by $O(1)$.

This is achieved by suitably batching work and appealing to monotone min-plus products [5, 10, 19]. The resulting algorithm is neither simple to implement nor practical, however. We describe our results in terms of *max-plus* matrix products as it matches the recurrences of Section 4. We write $\odot$ for max-plus matrix product:

$$(A \odot B)_{i,j} = \max_k \{A_{i,k} + B_{k,j}\}.$$

Lemma 9. *For fixed v and R, $q \mapsto RF[v, q, R]$ is nonincreasing. For fixed v and p, $L \mapsto LF[v, L, p]$ is nonincreasing.*

Proof: Increasing q shrinks the feasible set of intervals contained in $[q, R]$, and similarly increasing L shrinks the feasible set of intervals contained in $[L, p]$. Therefore the corresponding optimum value cannot increase. □

Theorem 2. *If all vertex weights are nonnegative integers bounded by $O(1)$, then a maximum weight induced forest of a circle graph can be computed deterministically in time $O(n^{(7+\omega)/2+o(1)})$ [5, 10, 19]. In particular, using $\omega < 2.371339$ [1], this is $O(n^{4.686})$.*

Proof: Since the vertex weights are bounded by $O(1)$, every feasible forest has total weight $O(n)$. Hence every entry of B, LF, RF, and every quantity of the form $w(v) + MR[u, v, p, R]$, is $O(n)$.

The products below are evaluated consistently with the dependencies of the recurrences. We process R in increasing order and, for a fixed R, order the nodes by decreasing right end-point. The term $RF[u, p + 1, q - 1]$ is already available because $q - 1 < R$, while $v \in RC(u, 0, R)$ implies $r_v > r_u$, so every same-R dependency of u precedes u in this order.

To keep the product dependencies acyclic, we apply a standard divide-and-conquer over the decreasing-right-end-point order. After the states indexed by the first half have been computed recursively, the relevant products batch all contributions from that half to the second half, after which we recurse on the second half. The matrices below describe one such batched update, with entries outside the corresponding two halves set to $-\infty$.

We first batch the computation of MR. For fixed u and R, define matrices $A^{u,R}$ and $B^{u,R}$ by

$$A^{u,R}_{p,q} = \begin{cases} RF[u, p+1, q-1], & \text{if } r_u + 1 \le q \le R \text{ and } p < q \\ -\infty & \text{otherwise} \end{cases}$$

and

$$B^{u,R}_{q,v} = \begin{cases} RF[v, q, R], & \text{if } v \in RC(u, 0, R) \text{ and } q \le r_v, \\ -\infty, & \text{otherwise} \end{cases}$$

By Lemma 9, each column of $B^{u,R}$ is nonincreasing, so $B^{u,R}$ is column-monotone. Moreover, for every valid v, p,

$$MR[u, v, p, R] = (A^{u,R} \odot B^{u,R})_{p,v}.$$

Thus the required MR values are obtained from square monotone max-plus products. Over all $O(n^2)$ choices of u, R, this costs

$$O(n^2 \cdot n^{(3+\omega)/2+o(1)}) = O(n^{(7+\omega)/2+o(1)})$$

time by [5, 19]. Next we batch the computation of RF. For fixed R, let

$$P_R = \{(v, p) : r_v \le R \text{ and } \ell_v \le p < r_v\}.$$

Then $|P_R| = O(n^2)$. Define matrices Y^R and X^R by

$$Y^R_{u,(v,p)} = \begin{cases} w(v) + MR[u, v, p, R], & \text{if } v \in RC(u, 0, R) \text{ and } \ell_v \le p < r_u \\ -\infty, & \text{otherwise} \end{cases}$$

and

$$X^R_{(v,p),L} = \begin{cases} LF[v, L, p], & \text{if } L \le \ell_v \le p < r_v \\ -\infty & \text{otherwise} \end{cases}$$

By Lemma 9, each row of X^R is nonincreasing, so X^R is row-monotone. Accumulating the contributions over all divide-and-conquer splits gives:

$$H^R_{u,L} = (Y^R \odot X^R)_{u,L} = \max_{v \in RC(u,L,R)} \max_{p \in [\ell_v, r_u - 1]} \big(w(v) + LF[v, L, p] + MR[u, v, p, R]\big).$$

Hence

$$RF[u, L, R] = \max\big(B[r_u + 1, R], H^R_{u,L}\big).$$

At each level of the divide-and-conquer, the required rectangular products, padded to the full dimensions, are $n \times n^2$ by $n^2 \times n$ monotone max-plus products. By [10, 19], this takes $O(n^{(1+2+1+\omega(2))/2+o(1)}) = O(n^{(4+\omega(2))/2+o(1)})$ time. Over all $O(n)$ choices of R, the total cost of computing all RF values is $O(n^{1+(4+\omega(2))/2+o(1)})$. Using the simple bound $\omega(2) \le \omega + 1$, this is at most $O(n^{(7+\omega)/2+o(1)})$. The table LF is computed symmetrically within the same bound.

Finally, all values of B and G still require only $O(n^3)$ time. Therefore the overall running time is $O(n^{(7+\omega)/2+o(1)})$ as claimed. □

Algorithm 1: `Compute-MWIF` in $O(n^3m)$ time, $O(n^3)$ space.

Input: Intervals $\{(\ell_v, r_v)\}_{v=1}^{n}$ with distinct endpoints in $[0, 2n-1]$, representing a circle graph G.
Output: The weight of a maximum induced forest in G, i.e. $B[0,\ 2n-1]$.

```
for R ← 0 to 2n − 1 do
    for e ← R to 0 do
        if ∃ an interval u with right end-point e then
            MR ← Compute-MR(u, R)
            Compute-RF(u, R, MR)
            Compute-B(u, R, RF)
        if ∃ an interval u left end-point e and r_u > R then
            ML ← Compute-ML(u, R)
            Compute-LF(u, R, ML)
return B[0, 2n − 1]
```

Algorithm 2: `Compute-MR`: Compute the matrix $MR[v,p]$ for a fixed (u,R).

Input: u, R; Intervals $\{(\ell_v, r_v)\}_{v=1}^{n}$, right children $RC[u]$; RF.
Output: $MR[v,p]$ for all $v \in RC[u]$ with $r_v \le R$ and $p \in [\ell_v,\ r_u - 1]$.

```
MR[v, p] ← 0 for all v, p
foreach v ∈ RC[u] with r_v ≤ R do
    for q ← r_u + 1 to r_v do
        for p ← ℓ_v to r_u − 1 do
            MR[v, p] ← max{MR[v, p], RF[v, q, R] + RF[u, p + 1, q − 1]}
return MR
```

6 Conclusion

We have described an algorithm for computing a maximum weight induced forest of a circle graph operating in $O(n^3m)$ time and $O(n^3)$ space, improving on Gavril's [14] $O(n^7)$ time algorithm, and also $O(n^3)$ space algorithm. The latter algorithm is defined in terms of computing a maximum weight independent set of a suitably defined interval-filament graph. The algorithm in this paper is arguably slightly simpler. In the unweighted case, this paper points out an $O(n^2mk)$ time algorithm, requiring $O(nk^2)$ space. We also note the existence of an $O(n^{4.686})$ time algorithm in the unweighted case, by utilising monotone min-plus products [5, 10, 19].

References

[1] J. Alman, R. Duan, V. V. Williams, Y. Xu, Z. Xu, and R. Zhou. More asymmetry yields faster matrix multiplication. In *Proceedings of the 2025 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2005–2039, 2025. URL: https://epubs.siam.org/doi/abs/10.1137/1.9781611978322.63, doi:10.1137/1.9781611978322.63.

Algorithm 3: `Compute-RF`: for fixed (u, R), fill all $RF[u, L, R]$ with $L \in (\ell_u, r_u]$.

Input: u, R; Intervals $\{(\ell_v, r_v)\}_{v=1}^{n}$; $RC[u]$; LF, RF, B; $MR[v, p]$ for this (u, R) from Algorithm 2.
Output: $RF[u, L, R]$ for all $L = \ell_u + 1, \ldots, r_u$.
for $L' \leftarrow \ell_u + 1$ **to** r_u **do**
  $RF[u, L', R] \leftarrow B[r_u + 1, R]$
foreach $v \in RC[u]$ ***with*** $\ell_v \leq r_u - 1$ ***and*** $r_v \leq R$ **do**
  for $p \leftarrow \ell_v$ **to** $r_u - 1$ **do**
    for $L' \leftarrow \ell_u + 1$ **to** ℓ_v **do**
      $RF[u, L', R] \leftarrow \max\{\, RF[u, L', R],\ w(v) + MR[v, p] + LF[v,\ L',\ p] \,\}$

Algorithm 4: `Compute-B`: Update $B[0, R], \ldots, B[\ell_u, R]$.

Input: Intervals $\{(\ell_v, r_v)\}_{v=1}^{n}$ u, R; RF.
Output: B with entries $B[0, R], \ldots B[\ell_u, R]$ updated.
$g \leftarrow 0$
for $s \leftarrow \ell_u$ **to** $r_u - 1$ **do**
  $g \leftarrow \max\{\, g,\ B[\ell_u + 1,\ s]\ +\ RF[u,\ s + 1,\ R]\,\}$
for $a \leftarrow 0$ **to** ℓ_u **do**
  $B[a, R] \leftarrow \max\{B[a, R],\ w(u) + g\}$

[2] M. Avriel, M. Penn, and N. Shpirer. Container ship stowage problem: complexity and connection to the coloring of circle graphs. *Discrete Applied Mathematics*, 103(1):271–279, 2000. doi:10.1016/S0166-218X(99)00245-0.

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

[4] A. Bouchet. Reducing prime graphs and recognizing circle graphs. *Combinatorica*, 7(3):243–254, Oct. 1987. doi:10.1007/BF02579301.

[5] S. Chi, R. Duan, T. Xie, and T. Zhang. Faster min-plus product for monotone instances. In *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2022, pages 1529–1542, New York, NY, USA, 2022. Association for Computing Machinery. doi:10.1145/3519935.3520057.

[6] J. K. H. Chiu and Y.-P. P. Chen. Efficient conversion of rna pseudoknots to knot-free structures using a graphical model. *IEEE Transactions on Biomedical Engineering*, 62(5):1265–1271, 2015. doi:10.1109/TBME.2014.2375360.

[7] J. Cong and C. L. Liu. Over-the-cell channel routing. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 9(4):408–418, 1990. doi:10.1109/43.45872.

[8] W. H. Cunningham. Decomposition of directed graphs. *Siam Journal on Algebraic and Discrete Methods*, 3:214–228, 1982. doi:10.1137/0603021.

[9] P. Damaschke. The hamiltonian circuit problem for circle graphs is np-complete. *Information Processing Letters*, 32(1):1–2, 1989. doi:10.1016/0020-0190(89)90059-8.

[10] A. Dürr. Improved bounds for rectangular monotone min-plus product and applications. *Information Processing Letters*, 181:106358, 2023. doi:10.1016/j.ipl.2023.106358.

[11] C. Eisenbeis, S. Lelait, and B. Marmol. The meeting graph: a new model for loop cyclic register allocation. In *Proceedings of the IFIP WG10.3 Working Conference on Parallel Architectures and Compilation Techniques*, PACT '95, pages 264–267, GBR, 1995. IFIP Working Group on Algol. URL: https://dl.acm.org/doi/10.5555/224659.224937.

[12] C. P. Gabor, K. J. Supowit, and W.-L. Hsu. Recognizing circle graphs in polynomial time. *J. ACM*, 36(3):435–473, July 1989. doi:10.1145/65950.65951.

[13] F. Gavril. Maximum weight independent sets and cliques in intersection graphs of filaments. *Information Processing Letters*, 73(5):181–188, 2000. URL: https://www.sciencedirect.com/science/article/pii/S0020019000000259, doi:10.1016/S0020-0190(00)00025-9.

[14] 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.

[15] F. Gavril. Minimum weight feedback vertex sets in circle n-gon graphs and circle trapezoid graphs. *Discrete Mathematics, Algorithms and Applications*, 03(03):323–336, 2011. doi:10.1142/S1793830911001243.

[16] E. Gioan, C. Paul, M. Tedder, and D. Corneil. Practical and efficient circle graph recognition. *Algorithmica*, 69(4):759–788, Aug. 2014. doi:10.1007/s00453-013-9745-8.

[17] M. C. Golumbic. Chapter 11 - not so perfect graphs. In M. C. Golumbic, editor, *Algorithmic Graph Theory and Perfect Graphs*, pages 235–253. Academic Press, 1980. doi:10.1016/B978-0-12-289260-8.50018-2.

[18] 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.

[19] C. Jin, J. Park, B. Saha, and Y. Xu. Deterministic Monotone Min-Plus Product and Convolution. In S. Bhattacharya, D. Nanongkai, M. Benedikt, and G. Puppis, editors, *53rd International Colloquium on Automata, Languages, and Programming (ICALP 2026)*, volume 374 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 119:1–119:23, Dagstuhl, Germany, 2026. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. URL: https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.ICALP.2026.119, doi:10.4230/LIPIcs.ICALP.2026.119.

[20] R. M. Karp. Reducibility among combinatorial problems. In R. E. Miller, J. W. Thatcher, and J. D. Bohlinger, editors, *Complexity of Computer Computations: Proceedings of a symposium on the Complexity of Computer Computations, held March 20–22, 1972, at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York, and sponsored by the Office of Naval Research, Mathematics Program, IBM World Trade Corporation, and the IBM Research Mathematical Sciences Department*, pages 85–103. Springer US, Boston, MA, 1972. doi:10.1007/978-1-4684-2001-2_9.

[21] J. M. Keil. The complexity of domination problems in circle graphs. *Discrete Applied Mathematics*, 42(1):51–63, 1993. URL: http://dx.doi.org/10.1016/0166-218X(93)90178-Q, doi:10.1016/0166-218X(93)90178-Q.

[22] W. Naji. Reconnaissance des graphes de cordes. *Discrete Mathematics*, 54(3):329–337, 1985. URL: https://www.sciencedirect.com/science/article/pii/0012365X85901177, doi:10.1016/0012-365X(85)90117-7.

[23] N. Nash and D. Gregg. An output sensitive algorithm for computing a maximum independent set of a circle graph. *Inf. Process. Lett.*, 110(16):630–634, 2010. URL: http://dx.doi.org/10.1016/j.ipl.2010.05.016, doi:10.1016/j.ipl.2010.05.016.

[24] 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.

[25] C. Paul and I. Rutter. Circle Graphs Can Be Recognized in Linear Time. In M. Mahajan, F. Manea, A. McIver, and N. K. Thang, editors, *43rd International Symposium on Theoretical Aspects of Computer Science (STACS 2026)*, volume 364 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 72:1–72:20, Dagstuhl, Germany, 2026. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. URL: https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.STACS.2026.72, doi:10.4230/LIPIcs.STACS.2026.72.

[26] J. Spinrad. Recognition of circle graphs. *Journal of Algorithms*, 16(2):264–282, 1994. URL: https://www.sciencedirect.com/science/article/pii/S0196677484710121, doi:10.1006/jagm.1994.1012.

[27] J. P. Spinrad. *Efficient graph representations / Jeremy P. Spinrad.* Fields Institute monographs, 19. American Mathematical Society, Providence, R.I, 2003.

[28] 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. doi:10.1109/TCAD.1987.1270250.

[29] A. Tiskin. Semi-local string comparison: algorithmic techniques and applications. *CoRR*, abs/0707.3619, 2013. URL: http://arxiv.org/abs/0707.3619, doi:10.48550/arXiv.0707.3619.

[30] W. Unger. On the k-colouring of circle-graphs. In *Proceedings of the 5th Annual Symposium on Theoretical Aspects of Computer Science*, STACS '88, pages 61–72, London, UK, UK, 1988. Springer-Verlag. doi:10.1007/BFb0035832.