Induced Disjoint Paths in Circular-Arc Graphs in Linear Time *

Petr A. Golovach[1], Daniël Paulusma[2], and Erik Jan van Leeuwen[3]

[1] Department of informatics, University of Bergen, Norway,
petr.golovach@ii.uib.no
[2] School of Engineering and Computer Science, Durham University, UK,
daniel.paulusma@durham.ac.uk
[3] Max-Planck Institut für Informatik, Saarbrücken, Germany,
erikjan@mpi-inf.mpg.de

Abstract. The Induced Disjoint Paths problem is to test whether a graph G with k distinct pairs of vertices (s_i, t_i) contains paths $P_1, \ldots, P_k$ such that P_i connects s_i and t_i for $i = 1, \ldots, k$, and P_i and P_j have neither common vertices nor adjacent vertices (except perhaps their ends) for $1 \leq i < j \leq k$. We present a linear-time algorithm that solves Induced Disjoint Paths and finds the corresponding paths (if they exist) on circular-arc graphs. For interval graphs, we exhibit a linear-time algorithm for the generalization of Induced Disjoint Paths where the pairs (s_i, t_i) are not necessarily distinct.

1 Introduction

A classic algorithmic problem on a graph G with k distinct pairs of vertices (s_i, t_i) is to find vertex-disjoint paths $P_1, \ldots, P_k$ such that P_i connects s_i and t_i for $i = 1, \ldots k$. Known as the Disjoint Paths problem, it is NP-complete on general graphs [15], but can be solved in $O(n^3)$ time for any fixed integer k [24] (i.e. it is fixed-parameter tractable). The Induced Disjoint Paths problem also takes as input a graph G with k distinct pairs of vertices (s_i, t_i) and also asks whether there are paths $P_1, \ldots, P_k$ such that P_i connects s_i and t_i for $i = 1, \ldots, k$, but with the extra condition that $P_1, \ldots, P_k$ must be *mutually induced*, that is, no two paths P_i, P_j have common or adjacent vertices (except perhaps their end-vertices). Notice that the Disjoint Paths problem can be reduced to Induced Disjoint Paths by subdividing every edge of the graph. The Induced Disjoint Paths problem is NP-complete even for instances with $k = 2$ [2, 5], and thus in particular is not fixed-parameter tractable unless P=NP.

The hardness of both Disjoint Paths and Induced Disjoint Paths on general graphs inspired research on their complexity on structured graph classes.

* This work is supported by EPSRC (EP/K025090/1) and Royal Society (JP100692). The research leading to these results has also received funding from the European Research Council under the European Union's Seventh Framework Programme (FP/2007-2013) / ERC Grant Agreement n. 267959.

On the negative side, Disjoint Paths remains NP-complete on line graphs [19] and split graphs [14]. Induced Disjoint Paths remains NP-complete on claw-free graphs [6] (in fact, even on line graphs). Both problems remain NP-complete on planar graphs [18, 8]. In these cases, however, fixed-parameter algorithms are known [9, 14, 16, 23, 24]. On the positive side, polynomial-time algorithms for Disjoint Paths exist on graphs of bounded treewidth [22] and graphs of cliquewidth at most 2 [12], and for Induced Disjoint Paths on AT-free graphs [8] and chordal graphs [1].

We focus on the complexity of Induced Disjoint Paths on circular-arc graphs. Recall that a *circular-arc graph* G has a *representation* in which each vertex of G corresponds to an arc of a circle, and two vertices of G are adjacent if and only if their corresponding arcs intersect. Circular-arc graphs generalize *interval graphs*, which have a representation in which each vertex corresponds to an interval of the line, and two vertices are adjacent if and only if their corresponding intervals intersect. The complexity of Disjoint Paths is known: it is NP-complete on interval graphs [21]. In contrast, for Induced Disjoint Paths, the authors of the present work recently showed a polynomial-time algorithm on circular-arc graphs [9] (for a weaker problem variant, such an algorithm is also implied by a general framework [7]). This work, as well as the polynomial-time algorithms on AT-free graphs [8] and chordal graphs [1], imply a polynomial-time algorithm on interval graphs. These algorithms, however, do not settle the complexity of Induced Disjoint Paths on circular-arc graphs (and interval graphs), as the question whether a linear-time algorithm exists is left open.

In this paper, we exhibit a linear-time algorithm for Induced Disjoint Paths on circular-arc graphs. This improves on the known algorithm for circular-arc graphs as well as the known algorithms for interval graphs. We also introduce a generalization of Induced Disjoint Paths called Requirement Induced Disjoint Paths, which is to find r_i paths that connect s_i and t_i for $i = 1, \ldots, k$, such that all paths are mutually induced. We present a linear-time algorithm for Requirement Induced Disjoint Paths on interval graphs. To solve these problems, our algorithms first preprocesses the instance. Some of the preprocessing rules build on our earlier work on Induced Disjoint Paths [8, 9], but care is required to adapt them for Requirement Induced Disjoint Paths and to execute them in linear time. Most preprocessing rules, however, are novel. After the preprocessing stage, the algorithms identify a set of candidate paths for each pair (s_i, t_i). For each candidate path for a pair (s_i, t_i), we add an arc with color i that corresponds to the path to an auxiliary graph H. Finally, we show that it suffices to find an independent set in H that contains r_i arcs of each color. We show that the algorithms perform all stages in linear time.

2 Preliminaries

We only consider finite undirected graphs that have no loops and no multiple edges. We refer to the textbook of Diestel [4] for any standard graph terminology not defined here. Let $G = (V, E)$ be a graph. For a set $S \subseteq V$, the graph $G[S]$

denotes the subgraph of G *induced by* S; that is, the graph with vertex set S and edge set $\{uv \in E \mid u, v \in S\}$. We write $G - S = G[V \setminus S]$. We denote the (open) neighborhood of a vertex u by $N_G(u) = \{v \mid uv \in E\}$ and its closed neighborhood by $N_G[u] = N_G(u) \cup \{u\}$. We denote the neighborhood of a set $U \subseteq V$ by $N_G(U) = \{v \in V \setminus U \mid uv \in E \text{ for some } u \in U\}$ and $N_G[U] = U \cup N_G(U)$. We denote the degree of a vertex u by $\deg_G(u) = |N_G(u)|$.

We denote an unordered pair of elements x, y by $\{x, y\}$ (i.e. $\{x, y\} = \{y, x\}$).

Problem Definition Let $P = v_1 \cdots v_r$ be a path (we call such a path a v_1v_r-*path*). The vertices v_1 and v_r are the *ends* or *end-vertices* of P, and the vertices $v_2, \ldots, v_{r-1}$ are the *inner vertices* of P. We say that an edge v_iv_j, $i + 1 < j$, is an *inner chord* of P if v_i or v_j is an inner vertex of P. Distinct paths $P_1, \ldots, P_\ell$ in a graph G are *mutually induced* if:

(i) each P_i has no inner chords;
(ii) any distinct P_i, P_j may only share vertices that are ends of both paths;
(iii) no inner vertex u of any P_i is adjacent to a vertex v of some P_j for $j \neq i$, except when v is an end-vertex of both P_i and P_j.

Notice that condition (i) may be assumed without loss of generality. This definition is more general than the definition in Section 1, as it allows the end-vertices of distinct paths to be the same or adjacent. We can now formally state our decision problem (where a *terminal* is some specified vertex).

Requirement Induced Disjoint Paths	
Instance:	a graph G, k pairs of distinct terminals $(s_1, t_1), \ldots, (s_k, t_k)$ such that $\{s_i, t_i\} \neq \{s_j, t_j\}$ for $0 \leq i < j \leq k$, and k positive integers $r_1, \ldots, r_k$.
Question:	does G have $\ell = r_1 + \ldots + r_k$ mutually induced paths $P_1, \ldots, P_\ell$ such that exactly r_i of these paths join s_i and t_i for $1 \leq i \leq k$?

If $r_1 = \ldots = r_k = 1$, then the problem is called Induced Disjoint Paths. The paths $P_1, \ldots, P_\ell$ are said to form a *solution* for a given instance, and we call every such path a *solution path*.

The problem definition allows a vertex v to be a terminal in two or more pairs (s_i, t_i) and (s_j, t_j). For instance, $v = s_i = s_j$ is possible. This corresponds to property (ii) of our definition of "being mutually induced". In order to avoid any confusion, we will view s_i and s_j as two different terminals "placed on" vertex v. Formally, we call v a *terminal vertex* that *represents* a terminal s_i or t_i if $v = s_i$ or $v = t_i$, respectively. We let T_v denote the set of terminals represented by v. If $T_v = \emptyset$, we call v a *non-terminal* vertex. We say that the two terminals s_i and t_i of a terminal pair (s_i, t_i) are *partners* of each other. If s_i is represented by u and t_i by v, then we also call a uv-path an s_it_i-path. By our problem definition, each terminal pair (s_i, t_i) consists of two distinct terminals. Hence, two partners are never represented by the same vertex.

By Property (i), each solution path P has no inner chords and P is an induced path if and only if its ends are non-adjacent. If two adjacent vertices u and v

represent terminals vertices belonging to the same pair (s_i, t_i), then the path uv is called a *terminal path* for s_i, t_i. We need the following observation.

Observation 1 *Any yes-instance of* Requirement Induced Disjoint Paths *has a solution that contains all possible terminal paths. In particular, a terminal path for a pair (s_i, t_i) is the unique $s_i t_i$-path in this solution if $r_i = 1$.*

Graph Classes Recall the definition of circular-arc and interval graphs from the introduction. Both graph types can be recognized in linear time and a corresponding representation can be found in linear time:

Theorem 1 ([3], see also [13, 17]). *An interval graph G with n vertices and m edges can be recognized in $O(n+m)$ time. In the same time, a representation of G can be constructed with interval end-points $1, \ldots, 2n$.*

Theorem 2 ([20]). *A circular-arc graph G with n vertices and m edges can be recognized in $O(n+m)$ time. In the same time, a representation of G can be constructed with arc end-points clockwise enumerated as $1, \ldots, 2n$.*

By Theorems 1 and 2, we always assume that an interval or circular-arc graph is given both by its adjacency list and its representation. Moreover, we assume that all the end-points of the intervals/arcs in the representation are distinct integers $1, \ldots, 2n$. Notice that using a representation we can check adjacency in $O(1)$ time. By slight abuse of notation, we often do not distinguish between the vertices and their corresponding intervals/arcs; e.g., we may speak of terminal intervals/arcs instead of terminal vertices.

For a vertex u of an interval graph, l_u and r_u denote the left and right end-point of u, respectively. Note that the degree of u is at least $(r_u - l_u - 1)/2$. For circular-arc graphs, we equate "left" to "counterclockwise" and "right" to "clockwise". Then, in the same way as for interval graphs, we let l_u and r_u denote the left and right end-point of a vertex u, respectively. In this way we are able to define similar terminology for both interval and circular-arc graphs. For two points x, y on the line, we write $x \leq y$ if y lies to the right with respect to x, and $x < y$ if $x \leq y$ and $x \neq y$, and we say that a point z *lies between* points x and y, if $x \leq z \leq y$. If x, y, z are points on a circle we write $x \leq z \leq y$ (or $x \leq z$ and $z \leq y$) to indicate that z is in the interval with the left end-point x and the right end-point y. We say that a vertex u *lies between* points x and y if $x \leq l_u < r_u \leq y$ (recall that l_u and r_u are distinct integers). Finally, a vertex u *lies between* two other vertices v, w if it lies between r_v and l_w; note that in that case we have in fact that $r_v < l_u < r_u < l_w$ by our assumption on the interval representation.

An *independent set* in a graph G is a set of vertices that are pairwise non-adjacent. At some stage, our algorithm for Induced Disjoint Paths on circular-arc graphs needs to compute a largest independent set of a circular-arc graph. This takes linear time:

Theorem 3 ([11]). *If the arc end-points of a circular-arc graph G are sorted, then a largest independent set of G can be found in $O(n)$ time.*

3 Interval Graphs

In this section we develop a linear-time algorithm that solves Requirement Induced Disjoint Paths on interval graphs.[1] A possible approach would be the following greedy algorithm: find a terminal vertex with the leftmost right end-point and trace path(s) for the corresponding terminal pairs by a greedy procedure that iteratively chooses the non-terminal vertex with the leftmost right end-point that does not conflict with vertices already chosen. However, we do not elaborate on this approach for two reasons. First, this approach would require a thorough case analysis (just like our algorithm, and thus not be substantially simpler). Second, and more importantly, the goal of this paper is to design a linear-time algorithm for Induced Disjoint Paths on circular-arc graphs, where we have no natural starting point for a similar greedy approach and guessing such a starting point would irrevocably lead to a quadratic-time algorithm.

We describe the main constructs of our algorithm. Consider an instance of Requirement Induced Disjoint Paths. Let P be an s_it_i-path that is not a terminal path, i.e. that has at least one inner vertex. Let I_P be the interval on the line obtained by taking the union of the intervals that correspond to the inner vertices of P. We say that P *covers* the interval I_P. Because P is an s_it_i-path, we say that I_P has color i.

Lemma 1. *Let $P_1, \ldots, P_\ell$ form a solution. The following statements hold:*

i) For $1 \leq i \leq k$, any interval I_{P_a} with color i intersects the intervals that represent s_i and t_i and does not intersect any other terminal interval;
ii) For $1 \leq a < b \leq \ell$, $I_{P_a} \cap I_{P_b} = \emptyset$;
iii) For $1 \leq i < j \leq k$, there is no interval with color j that lies between two intervals with color i, or vice versa.

We now outline our algorithm. Following Observation 1, we take all terminal paths into the solution. This might reduce the requirement r_i by 1 for some i. To find the remaining paths for all i, we determine a set of "candidate paths" that might or might not be used in the solution that we are constructing. The set of candidate paths is constructed such that for any s_it_i solution path P there is a candidate path P' such that P' is also an s_it_i-path and $I_{P'} \subseteq I_P$. We guarantee that the set of candidate paths has size $O(n)$. By Lemma 1, the paths that are selected in a solution must cover distinct parts of the line. Therefore, we create an auxiliary interval graph H that consists of all intervals covered by the candidate paths. The intervals covered by candidate s_it_i-paths all receive color i, for $i = 1, \ldots, k$. It then suffices to find an independent set with the required number of vertices of each color in H.

In the remainder of this section, we describe all steps of the algorithm in detail. We say that a step is *safe* if it runs in time $O(n + m + k)$ and is correct in the following sense:

[1] Due the space restrictions some proofs in this section and in the next ones are omitted or sketched. The full paper, with complete proofs, can be found in [10].

(i) a No-answer is given for no-instances only;
(ii) if a new instance is obtained, then it has a solution if and only if the original instance has a solution.
(iii) if a set of intervals that are all colored with color i is added to H, then this set has size $O(n)$ and corresponds to a *candidate set* of *candidate paths*.

The algorithm assumes that an interval representation of G is known, as given by Theorem 1. It also maintains an auxiliary interval graph H, initially empty. Recall that any vertex that we add to H will correspond to a candidate path for a solution. While adding vertices to H, we maintain an interval representation of H. Finally, the algorithm maintains a set $\mathcal{P}$ of paths, initially empty, which will form a solution for the instance (should it be a yes-instance). We let $T = \{s_1, t_1, \ldots, s_k, t_k\}$ be the set of all terminals. A terminal pair (s_i, t_i) is a *multi-pair* if $r_i \geq 2$, and a *simple pair* otherwise. The algorithm roughly consists of three stages: preprocess, construct H, and find an independent set.

3.1 Stage I: Preprocess

The only operations performed on G by our algorithm are vertex deletions. Hence, the graph that we obtain after each step is still interval. For simplicity, we denote this graph by G as well.

Step 1. Delete all non-terminal vertices that are adjacent to at least three terminal vertices.

Step 2. Check if there is a multi-pair that is represented by two non-adjacent terminal vertices. If so, then return a No-answer.

Lemma 2. *Steps 1 and 2 are safe.*

Suppose that we have not returned a No-answer after performing Step 2. In the next step, for each multi-pair, we identify a set of paths that together with the terminal paths form all candidate paths.

Step 3. For each non-terminal vertex u adjacent to terminal vertices v and w representing multi-pair terminals s_i and t_i, add I_{vuw} with color i to V_H, and delete u from G.

Lemma 3. *Step 3 is safe. Moreover, for any multi-pair (s_i, t_i), if P is a solution s_it_i-path with at least one inner vertex, then there is a candidate s_it_i-path P' with $I_{P'} \subseteq I_P$.*

In the next two steps, which are inspired by our earlier work on Induced Disjoint Paths [8, 9], we get rid of all adjacent terminal vertices that represent the same terminal pair. This includes (but is not limited to) all multi-pairs.

Step 4. Find the set Z of all terminal vertices v such that v only represents terminals whose partners are in $N_G(v)$. Delete the vertices of Z and all non-terminal vertices of $N_G(Z)$ from G. Delete from T the terminals of all terminal

pairs (s_i, t_i) with $s_i \in T_v$ or $t_i \in T_v$ for some $v \in Z$. Put all terminal paths corresponding to deleted terminal pairs in $\mathcal{P}$.

After Step 4, each terminal vertex represents at least one terminal whose partner is at distance at least 2. There may still be terminal pairs whose terminals are represented by adjacent vertices. We deal with such pairs in the next step.

Step 5. Delete all terminals s_i and t_i represented by adjacent terminal vertices from the terminal list, and delete all common non-terminal neighbors of the terminal vertices that represent s_i and t_i. Put all terminal paths corresponding to deleted terminals in $\mathcal{P}$.

Call a terminal pair *long* if its two terminals are represented by vertices of distance at least 2. After Step 5, all terminal pairs are long. Therefore, by Step 2, there are no multi-pairs anymore. Assume that there are $k' \leq k$ terminal pairs left; note that $k' = 0$ is possible.

Step 6. Check if there exists a terminal vertex that represents three or more terminals. If so, then return a `No`-answer.

After Step 6, a terminal vertex may represent at most two terminals (which must belong to different terminal pairs). We now observe that terminals should be ordered, and we let our algorithm find this ordering.

Step 7. Check if there exist three terminal vertices u, v, w such that u and w represent terminals from the same pair such that $l_u \leq l_v < l_w$. If so, then return a `No`-answer. Otherwise, order and rename the terminals such that $r_{u_i} < l_{v_i}$ and $l_{v_i} \leq l_{u_{i+1}}$ for $i = 1, \ldots, k' - 1$, where u_i, v_i are the vertices representing s_i, t_i, respectively.

Step 8. For $i \in \{1, \ldots, k' - 1\}$, if t_i and s_{i+1} are represented by distinct vertices u and v, delete all non-terminal vertices adjacent to both u and v.

Lemma 4. *Steps 4–8 are safe.*

3.2 Stage II: Construct H

We now construct the auxiliary H. Note that some intervals were already added to H as part of our preprocessing stage (see Step 3).

Step 9. For each $i \in \{1, \ldots, k'\}$, perform steps 9a–9d (where u and v are terminal vertices that represent s_i and t_i, respectively).

9a. For every common neighbor w of u and v, add the interval I_{uwv} to H with color i, and delete w from G.

9b. For each neighbor x of u not adjacent to v, determine whether there exists a neighbor y of v adjacent to x. If so, then choose y such that the right end-point of y is leftmost amongst all such neighbours of v. Add the interval I_{uxyv} to H with color i.

9c. Determine the connected components $C_1, \ldots, C_p$ of $G - (N[u] \cup N[v])$ whose vertices lie between r_u and l_v. For each C_j, determine the vertex $l(C_j)$ with the

leftmost left end-point and the vertex $r(C_j)$ with the rightmost right end-point. Then among the neighbors that $l(C_j)$ and u have in common, let $s_i(C_j)$ be the one with the rightmost left end-point (if it exists). Similarly, let $t_i(C_j)$ be the neighbor that $r(C_j)$ and v have in common and that has the leftmost right end-point (if it exists). Add the interval between the left end-point of $s_i(C_j)$ and the right end-point of $t_i(C_j)$ to H with color i, if it has not been added already in Step 9b (which might be the case if $s_i(C_j)$ and $t_i(C_j)$ intersect).

Lemma 5. *Step 9 is safe. Moreover, for $i = 1, \ldots, k'$, if P is a solution s_it_i-path, then there is a candidate s_it_i-path P' with $I_{P'} \subseteq I_P$.*

Proof. We first prove that Step 9 is correct. Let $i \in \{1, \ldots, k'\}$. Let u and v be the (non-adjacent) vertices of G representing s_i and t_i, respectively. Let P be a solution path for (s_i, t_i).

Suppose that P has length 2. Then P has exactly one inner vertex w, which is adjacent to both u and v. By Step 9a, H contains the interval I_P.

Suppose that P has length 3. Then P has exactly two inner vertices x and y' that are adjacent to u and v, respectively. Let y be the neighbor of v that is adjacent to x and has the leftmost right end-point among all such vertices. Then $P' = uxyv$ is an s_it_i-path. Notice that $I_{P'} \subseteq I_P$ by the choice of y and by the fact that u and v have no common neighbors after Step 9a. Therefore, in any solution that contains P, P can be replaced P'. By Step 9b, H contains $I_{P'}$.

Finally, suppose that P has length at least 4. Because P is an induced path, there is a connected component C_j of $G - (N[u] \cup N[v])$ whose vertices all lie between r_u and l_v, such that all inner vertices of P except two neighbors of u and v are in C_j. Let x' and y' be the neighbors of u and v on P, respectively. Let $x = s_i(C_j)$ and $y = t_i(C_j)$. Then from P we can construct an s_it_i-path P' by replacing x' and y' with x and y, respectively. Notice that $I_{P'} \subseteq I_P$ by the choice of y and by the fact that u and v have no common neighbors after Step 9a. Therefore, in any solution that contains P, P can be replaced P'. By Step 9c, H contains $I_{P'}$.

Observe that the above arguments prove that for $i = 1, \ldots, k'$, if P is a solution s_it_i-path, then there is a candidate s_it_i-path P' with $I_{P'} \subseteq I_P$.

We now show how to perform Step 9 in $O(n+m)$ time. In Step 9a, we add all the intervals that correspond to common neighbors of s_i and t_i for $i = 1, \ldots, k'$, and delete these common neighbors from G. Common neighbors of s_i and t_i are not common neighbors of terminals of any other pair by Step 8. Therefore, Step 9a takes $O(n + m)$ time in total, and $O(n)$ intervals are added to H. In Step 9b, for $i = 1, \ldots, k'$, we find for each neighbor x of s_i (recall that x is not adjacent to t_i after Step 9a), the neighbor y of t_i such that x and y are adjacent and the right end-point of y is leftmost. By using the adjacency lists for the neighbors of u, Step 9b takes $O(n+m)$ time in total, and $O(n)$ intervals are added to H. In Step 9c, we first find the connected components $C_1, \ldots, C_\ell$. This can be done by performing a breadth-first search. Because the connected components that we consider (and their vertices) are unique to a terminal pair, Step 9c takes $O(n+m)$ time in total. Again, $O(n)$ intervals are added to H. □

3.3 Stage III: Find Independent Set

It remains to find a particular independent set in H.

Step 10. Find an independent set in H that, for $i = 1, \ldots, k$, contains exactly $r_i - 1$ or r_i vertices colored i depending on whether (s_i, t_i) is a multi-pair or not. If such a set exists, add the corresponding candidate paths to $\mathcal{P}$ and return $\mathcal{P}$. Otherwise, return a `No`-answer.

Lemma 6. *Step 10 is safe.*

Proof. We first prove that Step 10 is correct. We do this by proving that our instance is a yes-instance if and only if H has an independent set as described in Step 10. First, suppose that H has such an independent set $\mathcal{I}$. For each interval u of color i, we can find an s_it_i-path in G with inner vertices that are used to construct u. Taking into account the terminal paths that are already included in $\mathcal{P}$, we obtain r_i s_it_i-paths for each $i \in \{1, \ldots, k\}$. We have to show that these paths are mutually induced. Because $\mathcal{I}$ is an independent set, distinct paths have no adjacent inner vertices. It remains to show that each $u \in \mathcal{I}$ does not intersect any terminal vertex (interval) of G except the vertices representing s_i, t_i. If u is added to H in Step 3, then it follows immediately from the fact that all non-terminal vertices that are adjacent to at least three terminals are deleted in Step 1 and from the description of Step 3. If u is added to H in Step 9, then notice u does not intersect any terminal vertex deleted in Step 4, because we delete them together with adjacent non-terminal vertices. Similarly, it does not interfere with any terminal deleted in Step 5, as proved in Lemma 4. Moreover, each interval added in Step 9 intersects exactly two remaining terminal vertices that are partners by Step 8. Hence, the instance is a yes-instance.

Now suppose that our instance is a yes-instance. Let $\ell_i = r_i - 1$ if (s_i, t_i) is a multi-pair, and let $\ell_i = r_i$ otherwise. By Observation 1, we can assume that the solution includes all terminal paths. Therefore, the solution contains exactly ℓ_i s_it_i-path with inner vertices. By Lemma 3 and Lemma 5, for each such solution s_it_i-path P, there is a candidate s_it_i path P' such that $I_{P'} \subseteq I_P$. Therefore, we can replace each solution path by a candidate path, and obtain a solution that uses only candidate paths. Let $\mathcal{I}$ denote the set of intervals covered by these paths. By Lemma 1, the intervals of $\mathcal{I}$ do not intersect each other. Moreover, by construction, $\mathcal{I}$ contains ℓ_i intervals with color i. Therefore, H has an independent set as described in Step 10.

We now show how to perform Step 10 in $O(n + m)$ time. We do this by performing the following procedure, which is a modification of the well-known greedy algorithm for finding a largest independent set in an interval graph.

1. Construct $2n$ buckets $L_1, \ldots, L_{2n}$ and $2n$ buckets $R_1, \ldots, R_{2n}$.

2. For each vertex u of H, put u in the buckets L_{l_u} and R_{r_u}.

3. Set $\mathcal{I} = \emptyset$ and $h = 2n$. For $i = 1, \ldots, k$, set $\ell_i = r_i - 1$ if (s_i, t_i) is a multi-pair, and set $\ell_i = r_i$ otherwise.

4. Scan the buckets $L_h, \ldots, L_1$ until we find a bucket L_j that contains a vertex u of H of some color i such that $\ell_i > 0$. Then u is included in $\mathcal{I}$. Find the set of vertices X from the buckets $R_j, \ldots, R_h$, and delete them from H. Then set $\ell_i = \ell_i - 1$, $h = j$, and repeat the procedure. We stop as soon as we cannot find the next bucket L_j.

If $\mathcal{I}$ contains less than ℓ_i vertices of color i for some $i \in \{1, \ldots, k\}$, then stop and return a `No`-answer. Otherwise, return $\mathcal{I}$. This procedure takes $O(|V(H)|) = O(n)$ time, and the corresponding paths can be found in $O(n+m)$ time. Hence, it remains to show that the procedure is correct. We need the following claim (proof omitted).

Claim 1. *Let U_i, U_j be the set of vertices (intervals) of H colored by distinct colors i and j respectively. Then for any $u \in U_i$ and $v \in U_j$, $l_u \neq l_v$. Moreover, if $l_u < l_v$ for some $u \in U_i$ and $v \in U_j$, then $l_x < l_y$ for any $x \in U_i$ and $y \in U_j$.*

Claim 1 implies that between the left endpoints of two intervals with a color i there can be no left endpoint of an interval with color $j \neq i$. Then, similar as the correctness of the well-known greedy algorithm for finding a largest independent set in an interval graphs, we can argue that the above procedure outputs the required independent set. □

As each step in our algorithm is safe, we obtain the following result.

Theorem 4. *The* Requirement Induced Disjoint Paths *problem can be solved in time $O(n+m+k)$ for interval graphs on n vertices and m edges with k terminal pairs.*

4 Circular-Arc Graphs

In this section, we modify the algorithm of the previous section to work for the Induced Disjoint Paths problem on circular-arc graphs. The general idea of the approach remains the same, but some preprocessing steps are no longer needed, and some steps need modification. In particular, we do not need colors here. We will again show that each step of the algorithm is *safe*, where the definition of a safe step remains the same, mutatis mutandis. The algorithm assumes that an arc representation of G is known, as given by Theorem 2. It maintains an auxiliary circular-arc graph H, initially empty, in a similar manner and function as before. It also maintains a set $\mathcal{P}$ of paths, initially empty.

The algorithm first performs Step 1. Note that Steps 2 and 3 are not necessary, as there are no multi-pairs now, and thus we do not apply them. We then continue with Steps 4 and 5.

Lemma 7. *Steps 1, 4, and 5 are safe.*

After Step 5, for each remaining terminal pairs (s_i, t_i), s_i and t_i are represented by vertices at distance at least two, and as before, we call such pairs *long*.

Let k' be the number of remaining terminal pairs. Notice that it can happen that $k' \leq 1$ after Step 5. It is convenient to handle this case separately.

Step 5$^+$. If $k' = 0$, then stop and return the solution $\mathcal{P}$. If $k' = 1$, then consider the terminal vertices u and v representing the terminals of the unique pair of T. Find a shortest uv-path P if it exists. If P exists, then add P to $\mathcal{P}$, and return the solution $\mathcal{P}$. Otherwise, stop and return a `No`-answer.

Lemma 8. *Step 5$^+$ is safe.*

Now we can assume that $k' \geq 2$. Since all pairs are long and $k' \geq 2$, there is only one direction around the circle that a solution path can go, and therefore, intuitively, the problem starts to behave roughly as it does on interval graphs. We perform Steps 6, 7, 8, and 9, where in Step 9 we do not color the vertices.

Lemma 9. *Steps 6, 7, 8, and 9 are safe. Moreover, for $i = 1, \ldots, k'$, if P is a solution s_it_i-path, then there is a candidate s_it_i-path P' with $I_{P'} \subseteq I_P$.*

Finally, we execute the following simplified version of Step 10.

Step 10*. Find a largest independent set in H using Theorem 3. If such a set exists, add the corresponding candidate paths to $\mathcal{P}$ and return $\mathcal{P}$. Otherwise, return a `No`-answer.

Lemma 10. *Step 10* is safe.*

As each step in our algorithm is safe, we obtain the following result.

Theorem 5. *The* Induced Disjoint Paths *problem can be solved in time $O(n + m + k)$ for circular-arc graphs on n vertices and m edges with k terminal pairs.*

5 Conclusion

We gave a linear-time algorithm for Requirement Induced Disjoint Paths on interval graphs, and for Induced Disjoint Paths on circular-arc graphs. By the application of the same ideas, we can solve Requirement Induced Disjoint Paths on n-vertex circular-arc graphs in time $O(n^2)$. The increase in running time is because to solve the auxiliary problem of finding a multicolored independent set we must "guess" a starting point for the greedy selection of such a set. As an aside, we can prove that finding a multicolored independent set is `NP`-complete when no order on the colors is given, even on interval graphs [10].

References

1. R. Belmonte, P.A. Golovach, P. Heggernes, P. van 't Hof, M. Kaminski and D. Paulusma, Detecting fixed patterns in chordal graphs in polynomial time. *Algorithmica* **69** (2014) 501–521.

2. D. Bienstock. On the complexity of testing for odd holes and induced odd paths. *Discrete Mathematics* **90** (1991) 85–92. See also Corrigendum, Discrete Mathematics **102** (1992) 109.
3. K.S. Booth and G.S. Lueker, Testing for the consecutive ones property, interval graphs, and graph planarity using PQ-tree algorithms. *J. Comput. Syst. Sci.* **13** (1976), 335–379.
4. R. Diestel, *Graph Theory*. Springer-Verlag, Electronic Edition, 2005.
5. M.R. Fellows. The Robertson–Seymour theorems: A survey of applications. *Proc. of the AMS-IMS-SIAM Joint Summer Research Conference*, Contemporary Mathematics, vol. 89, American Mathematical Society, Providence (1989) 1–18.
6. J. Fiala, M. Kamiński, B. Lidicky, and D. Paulusma. The k-in-a-path problem for claw-free graphs. *Algorithmica* **62** (2012) 499-519.
7. F.V. Fomin, I. Todinca, and Y. Villanger. Large induced subgraphs via triangulations and CMSO. In: *Proc. SODA 2014*, SIAM (2014) 582–593.
8. P.A. Golovach, D. Paulusma and E.J. van Leeuwen, Induced disjoint paths in AT-free graphs. In: *Proc. SWAT 2012*, LNCS 7357 (2012) 153–164.
9. P.A. Golovach, D. Paulusma and E.J. van Leeuwen, Induced disjoint paths in claw-free graphs. In: *Proc. ESA 2012*, LNCS 7501 (2012) 515–526.
10. P.A. Golovach, D. Paulusma and E.J. van Leeuwen, Induced disjoint paths in circular-arc graphs in linear time. CoRR abs/1403.0789 (2014).
11. M.C. Golumbic and P.L Hammer. Stability in circular arc graphs. *J. Algorithms* **9** (1988) 56–63.
12. F. Gurski, E. Wanke. Vertex disjoint paths on clique-width bounded graphs. *Theor. Comput. Sci.* **359** (2006) 188–199.
13. M. Habib, R.M. McConnell, C. Paul, and L. Viennot. Lex-BFS and partition refinement, with applications to transitive orientation, interval graph recognition and consecutive ones testing. *Theor. Comput. Sci.* **234**(2000) 59–84.
14. P. Heggernes, P. van 't Hof, R. Saei, E.J. van Leeuwen. Finding disjoint paths in split graphs. In: *Proc. SOFSEM 2014*, LNCS 8327 (2014) 315–326.
15. R.M. Karp. On the complexity of combinatorial problems. *Networks* **5** (1975) 45–68.
16. Y. Kobayashi and K. Kawarabayashi. A linear time algorithm for the induced disjoint paths problem in planar graphs. *J. Comput. Syst. Sci.* **78** (2012) 670–680.
17. N. Korte and R.H. Möhring. An incremental linear–time algorithm for recognizing interval graphs *SIAM J. Computing* **18** (1989) 68–81.
18. M. Kramer, J. van Leeuwen. The complexity of wirerouting and finding minimum area layouts for arbitrary VLSI circuits. *Adv. Comput. Res.* **2** (1984), 129–146.
19. J.F. Lynch. The equivalence of theorem proving and the interconnection problem. *SIGDA Newsletter* **5** (1975) 31–36.
20. R.M. McConnell. Linear-time recognition of circular-arc graphs. *Algorithmica* **37** (2003) 93–147.
21. S. Natarajan, A.P. Sprague. Disjoint paths in circular arc graphs. *Nordic J. Computing* **3** (1996) 256–270.
22. B.A. Reed. Tree width and tangles: A new connectivity measure and some applications. In: *Surveys in Combinatorics* Cambridge University Press, (1997) 87–162.
23. B.A. Reed, N. Robertson, A. Schrijver, P.D. Seymour. Finding disjoint trees in planar graphs in linear time. In: *Contemporary Mathematics* vol. 147, American Mathematical Society (1993) 295–301.
24. N. Robertson and P.D. Seymour. Graph minors. XIII. The disjoint paths problem. *Journal of Combinatorial Theory, Series B* **63** (1995) 65–110.