Theoretical Computer Science 640 (2016) 70–83

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Induced disjoint paths in circular-arc graphs in linear time ☆

Petr A. Golovach [a], Daniël Paulusma [b,*], Erik Jan van Leeuwen [c]

[a] Department of Informatics, University of Bergen, Norway
[b] School of Engineering and Computer Science, Durham University, UK
[c] Max-Planck-Institut für Informatik, Saarbrücken, Germany

ARTICLE INFO

Article history:
Received 13 February 2015
Received in revised form 14 May 2016
Accepted 6 June 2016
Available online 9 June 2016
Communicated by S. Sen

Keywords:
Induced disjoint paths
Circular-arc graphs
Linear-time algorithm

ABSTRACT

The Induced Disjoint Paths problem is to test whether an graph G on n vertices 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. In both cases, if a representation of the graph is given, then the algorithms run in $O(n+k)$ time.

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 [16], but can be solved in $O(n^3)$ time for any fixed integer k [25] (that is, 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. On the negative side, Disjoint Paths remains NP-complete on line graphs [20] and split graphs [14], and 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 [19,8]. In these cases, however, fixed-parameter algorithms are known [9, 14,17,24,25]. On the positive side, polynomial-time algorithms for Disjoint Paths exist on graphs of bounded treewidth [23] and graphs of clique-width 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. A *circular-arc* graph is a graph that has a *representation* in which each vertex corresponds to an arc of a circle, and two vertices are adjacent if and only if their

☆ 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 No. 267959. A preliminary version of this paper appeared as an extended abstract in the proceedings of WG 2014 [10].

* Corresponding author.
E-mail addresses: petr.golovach@ii.uib.no (P.A. Golovach), daniel.paulusma@durham.ac.uk (D. Paulusma), erikjan@mpi-inf.mpg.de (E.J. van Leeuwen).

http://dx.doi.org/10.1016/j.tcs.2016.06.003
0304-3975/

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 [22]. 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 do not settle the complexity of Induced Disjoint Paths on circular-arc graphs (and interval graphs) completely, 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. In both cases, if a representation of the graph is given and the graph has n vertices, then the algorithms run in $O(n+k)$ time.

Our two new algorithms first preprocess 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 $O(n+k)$ time on a graph on n vertices with k terminal pairs. Hence, most of our preprocessing rules 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 of 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 $O(n+k)$ 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 $G - S = G[V \setminus S]$. The (open) neighborhood and closed neighborhood of a vertex u are denoted by $N_G(u) = \{v \mid uv \in E\}$ and $N_G[u] = N_G(u) \cup \{u\}$, respectively. The open and closed neighborhoods of a set $U \subseteq V$ are denoted by $N_G(U) = \{v \in V \setminus U \mid uv \in E \text{ for some } u \in U\}$ and $N_G[U] = U \cup N_G(U)$, respectively. 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 are now able to 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