Induced Subgraph Isomorphism on proper interval and bipartite permutation graphs*

Pinar Heggernes† Pim van 't Hof† Daniel Meister‡ Yngve Villanger†

Abstract

Given two graphs G and H as input, the Induced Subgraph Isomorphism (ISI) problem is to decide whether G has an induced subgraph that is isomorphic to H. This problem is NP-complete already when G and H are restricted to disjoint unions of paths, and consequently also NP-complete on proper interval graphs and on bipartite permutation graphs. We show that ISI can be solved in polynomial time on proper interval graphs and on bipartite permutation graphs, provided that H is connected. As a consequence, we obtain that ISI is fixed-parameter tractable on these two graph classes, when parametrised by the number of connected components of H. Our results contrast and complement the following known results: $W[1]$-hardness of ISI on interval graphs when parametrised by the number of vertices of H, NP-completeness of ISI on connected interval graphs and on connected permutation graphs, and NP-completeness of Subgraph Isomorphism on connected proper interval graphs and connected bipartite permutation graphs.

1 Introduction

We study the following classical decision problem for undirected graphs, which has applications in a variety of practical areas [8, 9].

Induced Subgraph Isomorphism (ISI)
Input: Two graphs G and H.
Question: Does G have an induced subgraph isomorphic to H?

Equivalently, the question is whether we can delete vertices and their incident edges from G to obtain a graph isomorphic to H. ISI is a generalisation of several well-known problems like Clique, Independent Set, Longest Induced Path, and Graph Isomorphism. As a consequence of known hardness results on some of these problems, ISI is NP-complete [9] as well as $W[1]$-hard when parametrised by the number of vertices in H [7].

Due to its importance, ISI has been studied on a large number of graph classes in pursuit of polynomial-time solvability; a summary of such results is given in Figure 1. Unfortunately,

*This work is supported by the Research Council of Norway. A preliminary version of the result on proper interval graphs was presented at ISAAC 2010 [11].

†Department of Informatics, University of Bergen, P.O. Box 7803, N-5020 Bergen, Norway. Emails: {pinar.heggernes, pim.vanthof, yngve.villanger}@ii.uib.no

‡Theoretical Computer Science, University of Trier, Germany. Email: daniel.meister@uni-trier.de

the problem turns out to be notoriously difficult, and only very few non-trivial polynomial-time cases are known: ISI can be solved in polynomial time when G is a forest and H is a tree [17], when G and H are 2-connected outerplanar graphs [19], and when G is a trivially perfect graph and H is a threshold graph [1]. On the negative side, ISI remains NP-complete when G is a tree and H is a forest [9], when G is a cubic planar graph and H is a path [9], and when G and H are both connected cographs [5] or even connected trivially perfect graphs [1]. In fact, ISI is NP-complete already when G and H are disjoint unions of paths [9, 5]. This last NP-completeness result directly applies to all hereditary graph classes that contain arbitrarily long induced paths. In particular, ISI is NP-complete when G and H are proper interval graphs or when G and H are bipartite permutation graphs.

In this paper, we show that ISI can be solved in polynomial time when the input graphs are proper interval graphs and when they are bipartite permutation graphs, provided that the second input graph H is connected. In contrast, note that the same is not true for interval graphs or for permutation graphs. In particular, even if both input graphs are connected, ISI remains NP-complete on interval graphs [16] and on permutation graphs [5]. Another contrast is given by the fact that the related problem Subgraph Isomorphism, where one asks for a subgraph rather than an induced subgraph, is NP-complete when the two input graphs are both connected proper interval graphs or both connected bipartite permutation graphs [14].

To obtain our polynomial-time algorithms, we first solve the following problem. Given two graphs G and H, a vertex ordering σ_G and a colouring f_G for G, and a vertex ordering σ_H and a colouring f_H for H, decide whether there is an isomorphism between H and an induced subgraph of G that preserves the given vertex orderings and maps the vertices of H to vertices of G of the same colour. In this context, preserving the orderings means that the order of two vertices of H according to σ_H is the same as the order of their images in G according to σ_G. We show that if σ_G and f_G satisfy some specific conditions, then we can decide in polynomial time whether such an isomorphism exists. Our main results are then obtained by showing that proper interval graphs and bipartite permutation graphs have vertex orderings that satisfy these conditions.

Our results extend to showing that ISI is fixed-parameter tractable on proper interval graphs and on bipartite permutation graphs, when the parameter is the number of connected components of the second input graph H. This complements the results of [16], where it is shown that ISI is $W[1]$-hard on interval graphs when the parameter is the number of vertices of H, but it is fixed-parameter tractable on interval graphs when the parameter is the difference between the numbers of vertices of the two input graphs. To complete the list of known parametrised results on ISI, let us mention that when the parameter is the number of vertices of H, ISI is fixed-parameter tractable when both input graphs are planar [8] or their maximum vertex degree is bounded by a constant [3].

Our paper is organised as follows. Basic graph-theoretic definitions and notation are given in Section 2. We solve the above-mentioned problem on graphs with colourings and vertex orderings in Section 3. The main results on proper interval graphs and bipartite permutation graphs are given in Sections 4 and 5, respectively. We end with the fixed-parameter algorithms and concluding remarks in Section 6.

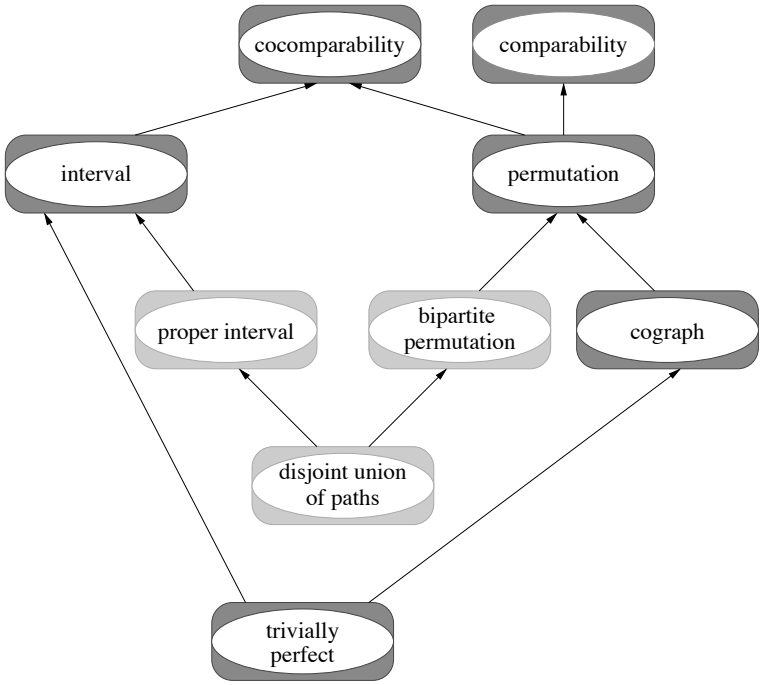

Figure 1: An overview of the graph classes mentioned in this paper and the complexity status of ISI on them. Here, "$\mathcal{C} \to \mathcal{D}$" is to be understood as "$\mathcal{C}$ is a subclass of $\mathcal{D}$". By the results of this paper, ISI is solvable in polynomial time on the graph classes with light grey frames, when H is connected. On the graph classes with dark grey frames, it is known from previous work that ISI is NP-complete even when both G and H are connected.

2 Definitions and notation

We consider simple finite undirected graphs. A graph G is an ordered pair (V, E), and $V = V(G)$ is the *vertex set* of G and $E = E(G)$ is the *edge set* of G. Edges of G are denoted as uv, where u and v are vertices of G. If uv is an edge of G then u and v are *adjacent* and u is a *neighbour* of v and v is a *neighbour* of u in G. Otherwise, if uv is not an edge of G, u and v are *non-adjacent* in G. The *(open) neighbourhood* of a vertex u in G is the set of the neighbours of u in G, and it is denoted as $N_G(u)$. The *closed neighbourhood* of u is $N_G[u] =_{\text{def}} N_G(u) \cup \{u\}$. For $X \subseteq V(G)$, the *subgraph of G induced by X*, denoted as $G[X]$, is the graph on vertex set X, and for every vertex pair u, v from X, uv is an edge of $G[X]$ if and only if uv is an edge of G. A graph G' is an *induced subgraph* of G if there exists a set $X \subseteq V(G)$ such that $G' = G[X]$. For a vertex u of G, we write $G-u$ to denote the induced subgraph $G[V(G) \setminus \{u\}]$ of G, that is, $G-u$ is the graph obtained from G by deleting vertex u. A set X of vertices of G is an *independent set* of G if the vertices in X are pairwise non-adjacent in G.

Let G be a graph. Let k be an integer with $k \geq 0$ and let u, v be a vertex pair of G. A *u, v-path of G of length k* is a sequence $(x_0, \ldots, x_k)$ of pairwise different vertices of G such that $x_0 = u$ and $x_k = v$ and $x_i x_{i+1} \in E(G)$ for every $0 \leq i < k$. Graph G is *connected* if G has a u, v-path for every vertex pair u, v of G; otherwise, if there is a vertex pair u, v of G such that G

has no u, v-path, G is called *disconnected*. A *connected component* of G is a maximal connected subgraph of G.

Let G and H be two graphs. We say that H is *isomorphic to* G if there is a bijective mapping φ from $V(H)$ to $V(G)$ such that for every vertex pair u, v of H, $uv \in E(H)$ if and only if $\varphi(u)\varphi(v) \in E(G)$. In this case, mapping φ is called an *isomorphism* from H to G. In Section 3, we will decide the existence of isomorphisms that satisfy additional conditions, namely that are required to respect vertex orderings and vertex colours. Let k be an integer with $k \geq 1$. A *k-colouring* for G is a mapping f that assigns a colour from the set $\{1, 2, \ldots, k\}$ to each vertex of G. A *colouring* for G is a k-colouring for G for some k. A colouring f for G is *proper* if for every edge uv of G, $f(u) \neq f(v)$. A *vertex ordering* for G is a linear arrangement $\sigma = \langle u_1, \ldots, u_n \rangle$ of the vertices of G. The *reverse* of σ is the vertex ordering $\langle u_n, \ldots, u_1 \rangle$. For a vertex pair x, y of G, we write $x \preccurlyeq_\sigma y$ if $x = u_i$ and $y = u_j$ for some indices i, j and $i \leq j$. If $x \neq y$ and thus $i < j$, we write $x \prec_\sigma y$. If $x \prec_\sigma y$, then we say that x *appears to the left of* y in the ordering σ, and y *appears to the right of* x in σ.

A graph class is *hereditary* if it is closed under taking induced subgraphs. All graph classes mentioned in this paper are hereditary [2, 10]. Furthermore, all graph classes in this paper admit well-known characterisations through vertex orderings, and we will often define graph classes using these characterisations. A graph G is a *comparability graph* if it has a vertex ordering σ such that for every vertex triple u, v, w of G with $u \prec_\sigma v \prec_\sigma w$, $uv \in E(G)$ and $vw \in E(G)$ implies $uw \in E(G)$ [10]. Such vertex orderings are called *comparability ordering*s. A graph is a *cocomparability graph* if its complement is a comparability graph. Equivalently, a graph G is a cocomparability graph if and only if it has a vertex ordering σ such that for every vertex triple u, v, w of G with $u \prec_\sigma v \prec_\sigma w$, $uw \in E(G)$ implies $uv \in E(G)$ or $vw \in E(G)$ [13]. Such vertex orderings are called *cocomparability ordering*s. A graph is *bipartite* if it has a proper 2-colouring, i.e., if its vertex set can be partitioned into two independent sets. Bipartite graphs are comparability graphs. Proper interval graphs and bipartite permutation graphs are both subclasses of cocomparability graphs [2, 10]; see also Figure 1. The definitions or characterisations, through vertex orderings, and necessary properties of proper interval graphs and of bipartite permutation graphs will be given at the beginnings of Sections 4 and 5. Comprehensive surveys on properties of the considered graph classes can be found in monographs such as [2] and [10].

Finally, for a brief background on parametrised complexity, a *parametrised problem* has a part of its input, typically an integer, identified as the parameter. A parametrised problem is *fixed-parameter tractable* if there is an algorithm for solving it in time $f(k) \cdot n^{\mathcal{O}(1)}$, where n is the total input size, k is the parameter, and f is a function that depends only on k and that does not involve n. There is a hierarchy of intractable parametrised problem classes [7]. For the results mentioned in this paper, it is sufficient to notice that a parametrised problem is unlikely to be fixed-parameter tractable if it is $W[1]$-hard.

3 ISI on graphs with colourings and vertex orderings

Let G and H be arbitrary graphs and let σ be a vertex ordering for G. Assume that G has an induced subgraph G' such that H is isomorphic to G'. Then, there are a vertex ordering τ for H and a total injective mapping $\varphi : V(H) \to V(G)$ such that the following two conditions are satisfied for every ordered vertex pair u, v of H:

1) $uv \in E(H)$ if and only if $\varphi(u)\varphi(v) \in E(G)$

2) $u \prec_\tau v$ if and only if $\varphi(u) \prec_\sigma \varphi(v)$.

The first condition is the isomorphism condition for H and G'. For the second condition, observe that τ can be obtained from σ by restriction to the vertices of G' and applying φ^{-1}. In this section, we will consider this type of an isomorphism problem.

The main result of this section is an efficient algorithm that decides the existence of an isomorphism that respects given colourings and vertex orderings. Let G and H be two graphs, and let σ and τ be vertex orderings for respectively G and H. Let $\varphi : V(H) \to V(G)$ be a total mapping. If for every ordered vertex pair u, v of H, $u \prec_\tau v$ implies $\varphi(u) \prec_\sigma \varphi(v)$ then we say that φ is (σ, τ)-*monotone*. Observe that a (σ, τ)-monotone mapping is injective. Assume that H is isomorphic to G, and let $\varphi : V(H) \to V(G)$ be an isomorphism from H to G. If φ is (σ, τ)-monotone, we say that φ is a (σ, τ)-*isomorphism* and that H is (σ, τ)-*isomorphic* to G. We say that H is (σ, τ)-*isomorphic to an induced subgraph* of G if there is an induced subgraph G' of G such that H is (σ', τ)-isomorphic to G', where σ' is the restriction of σ to the vertices of G'. Our algorithmic result of this section aims at deciding for given graphs G and H and vertex orderings σ and τ whether H is (σ, τ)-isomorphic to an induced subgraph of G.

In addition to the vertex ordering monotonicity of the desired isomorphism, we also ask for isomorphisms that map between equal classes of vertices. We formalise this class notion by colours. Let G and H be two graphs and let f_G and f_H be colourings for respectively G and H. Let $\varphi : V(H) \to V(G)$ be an arbitrary total mapping. We call φ *colour-preserving* for (G, f_G) and (H, f_H), if $f_H(x) = f_G(\varphi(x))$ for every vertex x of H. Our algorithm will decide the existence of colour-preserving isomorphisms.

Before we formally present our algorithm, we give an informal description. The algorithm receives as input two graphs G and H, as well as two colourings f_G and f_H and two vertex orderings σ and τ for respectively G and H. The algorithm decides whether H is colour-preserving (σ, τ)-isomorphic to an induced subgraph of G, which means that there is a (σ, τ)-isomorphism from H to an induced subgraph of G that is also colour-preserving. For the initial step, the algorithm maps the vertices of H to the rightmost possible vertices of G, only respecting the colourings and vertex orderings. We can say that the algorithm begins with the rightmost colour-preserving (σ, τ)-monotone mapping. Now, in rounds, the algorithm checks whether the current colour-preserving (σ, τ)-monotone mapping is also an isomorphism. If yes, the algorithm has found a colour-preserving (σ, τ)-monotone isomorphism from H to an induced subgraph of G. Otherwise, there is a "conflicting" vertex pair u, v of H that violates the isomorphism condition, which means that u and v are adjacent in H while their images are non-adjacent in G, or vice-versa. Based on these two possible cases, the algorithm computes a new colour-preserving (σ, τ)-monotone mapping by pushing u or v further left, and enters a new round.

The crucial auxiliary operation is the push operation for a vertex. Intuitively, the push operation applied to a vertex u finds the closest vertex to the left of u of a specified colour, if a vertex of that colour exists. We base the push operation on the "predecessor" function. Let G be a graph, let f be a colouring for G and let $\sigma = \langle x_1, \ldots, x_n \rangle$ be a vertex ordering for G. Let a be a vertex position index with $1 \leq a \leq n$ and let c be a colour. Then,

$$\mathrm{pred}_{(\sigma,f)}(a, c) \ =_{\mathrm{def}} \ \max\Big(\big\{ a' : 1 \leq a' < a \text{ and } f(x_{a'}) = c \big\} \cup \big\{ 0 \big\} \Big) .$$

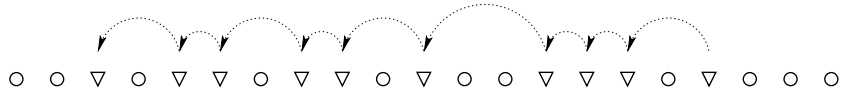

Figure 2: Depicted is a vertex ordering for a graph. The vertices of the graph are coloured with the colours "triangle" and "circle". We iteratively apply the predecessor function pred, starting at the rightmost vertex of colour "triangle", and it lists all vertices with colour "triangle", in the order of appearance in the vertex ordering, from right to left.

In words, when given as input a vertex position a and a colour c, the predecessor function $\text{pred}_{(\sigma,f)}$ considers all vertices to the left of x_a in vertex ordering σ that have colour c and it outputs the largest such position, unless no vertex to the left of x_a in σ has colour c, in which case it outputs 0. Note that x_a itself does not need to have colour c. As an example for the iterated application of pred to a graph with a 2-colouring, Figure 2 indicates the predecessor of each triangle-coloured vertex.

We are now ready to present the algorithm. It is called OrderedInducedSubgraph, and we often use OIS for short. It receives as input a 6-tuple $(G, f_G, \sigma;\ H, f_H, \tau)$ of the following components:

- G is a graph, f_G is a colouring for G, and σ is a vertex ordering for G
- H is a graph, f_H is a colouring for H, and τ is a vertex ordering of H.

The algorithm decides whether H is colour-preserving (σ, τ)-isomorphic to an induced subgraph of G.

Algorithm OrderedInducedSubgraph (OIS)

Input A graph G with colouring f_G and vertex ordering $\sigma = \langle x_1, \ldots, x_n \rangle$;
a graph H with colouring f_H and vertex ordering $\tau = \langle y_1, \ldots, y_r \rangle$.

begin
if $f_G(x_i) \neq f_H(y_r)$ for every $1 \leq i \leq n$ **then** **reject** **end if**;
let a_r be the largest vertex position index such that $f_G(x_{a_r}) = f_H(y_r)$;
for $i = r - 1$ **downto** 1 **do** **let** $a_i = \text{pred}_{(\sigma, f_G)}(a_{i+1}, f_H(y_i))$ **end for**;
if $a_1 = 0$ **then** **reject** **end if**;
while H is not (σ, τ)-isomorphic to $G[\{x_{a_1}, \ldots, x_{a_r}\}]$ **do**
let i, j be a vertex position index pair with $1 \leq i < j \leq r$
such that $y_i y_j \notin E(H)$ if and only if $x_{a_i} x_{a_j} \in E(G)$;
if $y_i y_j \notin E(H)$ **then**
PUSH(i)
else
PUSH(j)
end if
end while;
return $(a_1, \ldots, a_r)$ and **accept**
end.

Subroutine PUSH(b)
begin
set $a_b = \mathrm{pred}_{(\sigma,f_G)}(a_b, f_H(y_b))$;
while $a_b \leq a_{b-1}$ **and** $b \geq 2$ **do**
set $b = b - 1$;
set $a_b = \mathrm{pred}_{(\sigma,f_G)}(a_{b+1}, f_H(y_b))$
end while;
if $a_1 = 0$ **then reject end if**
end.

We analyse the correctness of the algorithm in two steps: we show first that the algorithm always accepts correctly, and then that it always rejects correctly. The algorithm consists of a main procedure and a subroutine called PUSH. The main procedure iteratively computes a sequence $\langle \mathfrak{a}^0, \ldots, \mathfrak{a}^h \rangle$ of vertex position index tuples, and each such tuple is of the form $(a_1, \ldots, a_r)$. Note here that input graph H has r vertices. The **for** loop of the main procedure computes the initialising tuple $\mathfrak{a}^0$, and the **while** loop computes the rest of the tuples $\mathfrak{a}^1, \ldots, \mathfrak{a}^h$. Note that only tuple $\mathfrak{a}^h$ is output, and this is done only if the algorithm accepts. An execution of the **while** loop body of the main procedure is called a *round*. For an integer $e \geq 0$, the index tuple $\mathfrak{a}^e = (a_1^e, \ldots, a_r^e)$ is the result of round e, and the values of $a_1^e, \ldots, a_r^e$ are the values of respectively $a_1, \ldots, a_r$ of the algorithm at the end of round e, in particular, after the application of Subroutine PUSH. The execution of the **for** loop is equivalent to round 0. For two r-tuples $\mathfrak{b} = (b_1, \ldots, b_r)$ and $\mathfrak{c} = (c_1, \ldots, c_r)$, we write $\mathfrak{b} \leq \mathfrak{c}$ if $b_i \leq c_i$ for every $1 \leq i \leq r$, and we write $\mathfrak{b} < \mathfrak{c}$ if $\mathfrak{b} \leq \mathfrak{c}$ and $b_i < c_i$ for some $1 \leq i \leq r$.

Lemma 3.1. *Let G be a graph with colouring f_G and vertex ordering $\sigma = \langle x_1, \ldots, x_n \rangle$, and let H be a graph with colouring f_H and vertex ordering $\tau = \langle y_1, \ldots, y_r \rangle$. When Algorithm* OIS *is run on input $(G, f_G, \sigma;\ H, f_H, \tau)$, the following holds:*

1) *for every $0 \leq e \leq h$, where $\mathfrak{a}^e = (a_1^e, \ldots, a_r^e)$: if $a_1^e \geq 1$ then $a_1^e < \cdots < a_r^e$*
2) *for every $0 \leq e \leq h$, where $\mathfrak{a}^e = (a_1^e, \ldots, a_r^e)$: if $a_1^e \geq 1$ then $f_H(y_i) = f_G(x_{a_i^e})$ for every $1 \leq i \leq r$*
3) *for every $1 \leq e \leq h$: $\mathfrak{a}^e \leq \mathfrak{a}^{e-1}$*
4) *if* OIS *accepts on input $(G, f_G, \sigma;\ H, f_H, \tau)$ and outputs $(a_1, \ldots, a_r)$ then H is colour-preserving (σ, τ)-isomorphic to $G[\{x_{a_1}, \ldots, x_{a_r}\}]$.*

Proof. Let $\langle \mathfrak{a}^0, \ldots, \mathfrak{a}^h \rangle$ be the computed vertex position index tuple sequence.

We prove statements 1 and 2 simultaneously, by induction on e. Let $e = 0$, and assume that $a_1^0 \geq 1$. Then, $f_H(y_r) = f_G(x_{a_r^0})$ due to the choice of a_r^0, and $a_i^0 < a_{i+1}^0$ and $f_G(x_{a_i^0}) = f_H(y_i)$ for every $1 \leq i < r$, according to the definition of the predecessor function pred. Thus, $\mathfrak{a}^0$ satisfies the two conditions.

Now, let $e \geq 1$, assume that $a_1^e \geq 1$, and assume that $\mathfrak{a}^{e-1}$ satisfies the two conditions. The new index tuple $\mathfrak{a}^e$ is computed during an execution of the **while** loop body, thus, it is the result of an execution of Subroutine PUSH. Since $a_1^{e-1} < \cdots < a_r^{e-1}$ by assumption,

the iterated application of the predecessor function in Subroutine PUSH yields $a_1^e < \cdots < a_r^e$. It is crucial to recall our assumption about $a_1^e \geq 1$ here. The same assumption also implies $f_G(x_{a_i^e}) = f_G(x_{a_i^{e-1}}) = f_H(y_i)$ for every $1 \leq i \leq r$, which is due to the definition of pred and our assumption about $\mathfrak{a}^{e-1}$. Thus, $\mathfrak{a}^e$ indeed satisfies the two conditions.

Statement 3 is a direct consequence of the definition of Subroutine PUSH and the properties of the predecessor function pred.

We prove statement 4. Assume that OIS accepts on input $(G, f_G, \sigma;\ H, f_H, \tau)$. Observe that OIS accepts only in the main procedure, and the initialising step must have been executed successfully, in particular, $a_1^0 \geq 1$. It follows that round h is the last fully executed round of the algorithm, and the condition of the **while** loop of the main procedure is false at the beginning of round $h+1$. This means that $a_1^h \geq 1$, since $a_1^h = 0$ and $h \geq 1$ means that the algorithm would reject during the execution of Subroutine PUSH in round h.

So, the condition of the **while** loop is not satisfied, which means that H is (σ, τ)-isomorphic to $G[\{x_{a_1^h}, \ldots, x_{a_r^h}\}]$. According to the results of statements 1 and 2, it directly follows that H is colour-preserving (σ, τ)-isomorphic to $G[\{x_{a_1^h}, \ldots, x_{a_r^h}\}]$, which proves statement 4. ∎

The main result of Lemma 3.1 is statement 4, that the algorithm always accepts correctly. For the second step toward a correctness proof for the algorithm, we want to show that the algorithm also always rejects correctly. Before we proceed with this step, we make some general observations. Let G and H be two graphs, and let σ and τ be vertex orderings for respectively G and H. Observe that it can be decided in polynomial time whether H is (σ, τ)-isomorphic to G, since only one mapping from $V(H)$ to $V(G)$ is possible. On the other hand, when generalising the problem to induced subgraphs, deciding whether H is (σ, τ)-isomorphic to an induced subgraph of G is NP-complete on general graphs. A straightforward reduction can be constructed from CLIQUE. And to fully satisfy the setting of our special isomorphism problem, we can additionally equip G and H with a 1-colouring. As a consequence, we cannot expect our algorithm to work correctly on all possible inputs. We restrict our inputs and require them to satisfy two special conditions. These two conditions are defined next.

Definition 3.2. *Let G be a graph, let f be a colouring for G, and let σ be a vertex ordering for G.*

1) *(G, f, σ) satisfies the* left umbrella condition *if for every vertex triple u, v, w of G with $u \prec_\sigma v \prec_\sigma w$ and $f(v) = f(w)$, $uw \in E(G)$ implies $uv \in E(G)$.*

2) *(G, f, σ) satisfies the* right umbrella condition *if for every vertex triple u, v, w of G with $u \prec_\sigma v \prec_\sigma w$ and $f(u) = f(v)$, $uw \in E(G)$ implies $vw \in E(G)$.*

Observe that the two umbrella conditions are symmetric, which means that (G, f, σ) satisfies the left umbrella condition if and only if (G, f, σ^R) satisfies the right umbrella condition, where σ^R is the reverse of σ. An illustrating description of the two umbrella conditions is given in Figure 3.

Lemma 3.3. *Let G be a graph with colouring f_G and vertex ordering $\sigma = \langle x_1, \ldots, x_n \rangle$ such that (G, f_G, σ) satisfies the left and right umbrella conditions, and let H be a graph with colouring f_H and vertex ordering $\tau = \langle y_1, \ldots, y_r \rangle$. If Algorithm* OIS *rejects input $(G, f_G, \sigma;\ H, f_H, \tau)$ then H is not colour-preserving (σ, τ)-isomorphic to any induced subgraph of G.*

left umbrella condition right umbrella condition

Figure 3: An illustration of the left and right umbrella conditions of Definition 3.2. The vertices u, v, w satisfy $u \prec_\sigma v \prec_\sigma w$, and we have vertices of two colours, namely black and white.

Proof. Recall the definitions preceding Lemma 3.1 and the technical results from Lemma 3.1; we will make heavy use of these throughout the proof. Let $\langle \mathfrak{a}^0, \ldots, \mathfrak{a}^h \rangle$ be the computed index tuple sequence. We prove the contraposition of the statement of the lemma. Let $\sigma = \langle x_1, \ldots, x_n \rangle$ and $\tau = \langle y_1, \ldots, y_r \rangle$.

We assume that H is colour-preserving (σ, τ)-isomorphic to an induced subgraph of G. Then, there are indices $d_1, \ldots, d_r$ with $1 \leq d_1 < \cdots < d_r \leq n$ such that H is colour-preserving (σ, τ)-isomorphic to $G[\{x_{d_1}, \ldots, x_{d_r}\}]$. Recall that this particularly means $f_H(y_i) = f_G(x_{d_i})$ for every $1 \leq i \leq r$. We let $\mathfrak{d} =_{\text{def}} (d_1, \ldots, d_r)$. It will be important later that

$$d_i \;\leq\; \text{pred}_{(\sigma, f_G)}(d_{i+1}, f_H(y_i))$$

holds for every $1 \leq i < r$. We show that OIS accepts with output tuple $\mathfrak{a}^h$ and that $\mathfrak{d} \leq \mathfrak{a}^h$ holds. To prove the result, we show by induction on e that $\mathfrak{d} \leq \mathfrak{a}^e$ holds.

Induction base
Recall the choice of $\mathfrak{d}$ and the definition of $\mathfrak{a}^0$, and observe that the following holds:

$$f_G(x_{d_r}) \;=\; f_G(x_{a_r^0}) \quad \text{and} \quad d_r \;\leq\; a_r^0 \,.$$

We consider the other entries of $\mathfrak{d}$ and $\mathfrak{a}^0$. Let $1 \leq i < r$. Recall from the initialising step of OIS:

$$a_i^0 \;=\; \text{pred}_{(\sigma, f_G)}(a_{i+1}^0, f_H(y_i)) \,.$$

It follows that $a_i^0 < a_{i+1}^0$ and $f_G(x_j) \neq f_H(y_i)$, for every $a_i^0 < j < a_{i+1}^0$. Since $d_i < d_{i+1} \leq a_{i+1}^0$ and $f_G(x_{d_i}) = f_H(y_i)$, it follows that $d_i \leq a_i^0$. We conclude $\mathfrak{d} \leq \mathfrak{a}^0$, which proves the induction base.

Induction step
We consider an arbitrary round e, for $0 \leq e < h$. We assume $\mathfrak{d} \leq \mathfrak{a}^e$, and we show $\mathfrak{d} \leq \mathfrak{a}^{e+1}$. Suppose for a contradiction that $\mathfrak{d} \not\leq \mathfrak{a}^{e+1}$. Then, there exists an index t with $1 \leq t \leq r$ such that $a_t^{e+1} < d_t$. Due to the assumption $\mathfrak{d} \leq \mathfrak{a}^e$, this means $a_t^{e+1} < d_t \leq a_t^e$.

We make some preparations. Recall that $\mathfrak{a}^{e+1}$ is obtained from $\mathfrak{a}^e$ by an application of Subroutine PUSH. Let p, q be the conflicting index pair chosen at the beginning of round $e+1$, and we can assume $p < q$. According to the choice of p and q, it holds: $y_p y_q \notin E(H)$ if and only if $x_{a_p^e} x_{a_q^e} \in E(G)$. Recall that PUSH is invoked with p or q. Let b be the number PUSH is invoked with. The following is a simple observation from the definition of PUSH: $a_i^{e+1} = a_i^e$ for every $b < i \leq r$. As an as simple as important consequence: $t \leq b$. Furthermore, according to PUSH:

$$a_i^{e+1} \;=\; \text{pred}_{(\sigma, f_G)}(a_{i+1}^{e+1}, f_H(y_i))$$

for every $t \leq i < b$.

We consider the values of $d_t, \ldots, d_b$ and $a_t^{e+1}, \ldots, a_b^{e+1}$. Let i be an index with $t \leq i < b$, and assume $d_{i+1} \leq a_{i+1}^{e+1}$. The monotonicity of the predecessor function $\mathrm{pred}_{(\sigma,f_G)}$ shows:

$$\mathrm{pred}_{(\sigma,f_G)}(d_{i+1}, f_H(y_i)) \;\leq\; \mathrm{pred}_{(\sigma,f_G)}(a_{i+1}^{e+1}, f_H(y_i))\,,$$

which implies

$$d_i \;\leq\; \mathrm{pred}_{(\sigma,f_G)}(d_{i+1}, f_H(y_i)) \;\leq\; \mathrm{pred}_{(\sigma,f_G)}(a_{i+1}^{e+1}, f_H(y_i)) \;=\; a_i^{e+1}\,,$$

so that $d_{i+1} \leq a_{i+1}^{e+1}$ implies $d_i \leq a_i^{e+1}$. From this we conclude that if $d_b \leq a_b^{e+1}$ then $d_t \leq a_t^{e+1}$. Since $a_t^{e+1} < d_t$ according to our assumptions, we can conclude $a_b^{e+1} < d_b$, and thus, $a_b^{e+1} < d_b \leq a_b^e$. Furthermore, since

$$f_G(x_{a_b^{e+1}}) \;=\; f_G(x_{d_b}) \;=\; f_G(x_{a_b^e}) \;=\; f_H(y_b) \quad\text{and}\quad a_b^{e+1} \;=\; \mathrm{pred}_{(\sigma,f_G)}(a_b^e, f_H(y_b))\,,$$

we conclude $d_b = a_b^e$. Informally, this means that $x_{a_b^e}$ is a correct choice for y_b.

We employ $d_b = a_b^e$ to construct the desired contradiction. We distinguish between the two cases about the value of b: either $b = p$ or $b = q$.

Case 1. PUSH is invoked with parameter value $b = p$.

Proof of the case. We have the following situation:

$$y_p y_q \notin E(H); \quad x_{d_p} x_{d_q} \notin E(G); \quad x_{a_p^e} x_{a_q^e} \in E(G); \quad d_p = a_p^e; \quad p = b < q\,.$$

Recall from the above that $b = p$ and $d_b = a_b^e$ implies $d_p = a_p^e$. Since $a_p^e = d_p < d_q \leq a_q^e$ and $x_{d_p} x_{d_q} \notin E(G)$ and $x_{d_p} x_{a_q^e} \in E(G)$, it clearly follows $d_q \neq a_q^e$, so that $a_p^e = d_p < d_q < a_q^e$ must hold.

We construct the desired contradiction. Observe that $f_G(x_{d_q}) = f_G(x_{a_q^e})$. Since $x_{a_p^e} x_{a_q^e} \in E(G)$ and since (G, f_G, σ) satisfies the left umbrella condition, it follows that $x_{d_p} x_{d_q} \in E(G)$ must hold, contradicting the observed situation. □

Case 2. PUSH is invoked with parameter value $b = q$.

Proof of the case. We have the following situation:

$$y_p y_q \in E(H); \quad x_{d_p} x_{d_q} \in E(G); \quad x_{a_p^e} x_{a_q^e} \notin E(G); \quad d_q = a_q^e; \quad p < q = b\,.$$

Recall from the above that $b = q$ and $d_b = a_b^e$ implies $d_q = a_q^e$. Since $d_p \leq a_p^e < a_q^e = d_q$ and $x_{d_p} x_{d_q} \in E(G)$ and $x_{a_p^e} x_{d_q} \notin E(G)$, it clearly follows $d_p \neq a_p^e$, so that $d_p < a_p^e < a_q^e = d_q$ must hold.

Since $x_{a_p^e} x_{a_q^e} \notin E(G)$ and $f_G(x_{d_p}) = f_G(x_{a_p^e})$ and since (G, f_G, σ) satisfies the right umbrella condition, it follows that $x_{d_p} x_{d_q} \notin E(G)$, which gives the desired contradiction. □

We summarise the shown results. We have seen that $\mathfrak{d} \leq \mathfrak{a}^h \leq \cdots \leq \mathfrak{a}^0$ holds. And since $\mathfrak{a}^{e+1} \neq \mathfrak{a}^e$ according to the definition of Subroutine PUSH, it even holds that $\mathfrak{d} \leq \mathfrak{a}^h < \cdots < \mathfrak{a}^0$. We conclude that OIS does not reject. Since OIS clearly terminates on each input, we conclude that OIS accepts. ■

Corollary 3.4. *Let G be a graph with colouring f_G and vertex ordering $\sigma = \langle x_1, \ldots, x_n \rangle$ such that (G, f_G, σ) satisfies the left and right umbrella conditions, and let H be a graph with colouring f_H and vertex ordering $\tau = \langle y_1, \ldots, y_r \rangle$. Algorithm* OIS *on input $(G, f_G, \sigma;\ H, f_H, \tau)$ decides in polynomial time whether H is colour-preserving (σ, τ)-isomorphic to an induced subgraph of G.*

Proof. The correctness of the algorithm is the joined result of statement 4 of Lemma 3.1 and of Lemma 3.3.

For the running time, recall the end of the proof of Lemma 3.3. It is clear that each round of the algorithm has polynomial running time, so it remains to bound the number of rounds, which is parameter h. We may assume that the algorithm output a tuple $\mathfrak{a}^h$. As argued in the last paragraph of the proof of Lemma 3.3, it holds that $\mathfrak{a}^h < \cdots < \mathfrak{a}^0$, which implies

$$\sum_{j=1}^{r} a_j^h \ < \ \sum_{j=1}^{r} a_j^{h-1} \ < \cdots < \ \sum_{j=1}^{r} a_j^0 \,. \tag{1}$$

Since the algorithm did not reject, it holds that $1 \leq a_i^j \leq n$ for every $1 \leq i \leq r$ and $0 \leq j < h$, where n and r are the numbers of vertices of respectively G and H. In particular, this means $r \leq \sum_{j=1}^{r} a_j^h$ and $\sum_{j=1}^{r} a_j^0 \leq rn$. This, together with (1), implies $r \leq rn - h$, so that we can conclude $h \leq r(n-1) \leq n^2$ about the number of executed rounds. ∎

Before we end this section, we review the obtained results. We showed that OIS correctly decides the existence of colour-preserving monotone induced subgraph isomorphisms, provided the input satisfies two conditions. These two conditions are the left and right umbrella conditions. At first glance, this appears to be a strong restriction on the possible inputs. At second glance, such restrictions are inavoidable, as we already discussed in this section. We will apply OIS to solve the induced subgraph isomorphism problem on proper interval graphs and bipartite permutation graphs. And for these graph classes, we can choose inputs that in fact satisfy the two conditions naturally.

We consider OIS more closely. If OIS accepts then it also outputs a vertex position index tuple, namely $\mathfrak{a}^h$, which defines a colour-preserving monotone induced subgraph isomorphism, according to statement 4 of Lemma 3.1. Let $\mathfrak{d}$ be an index tuple that defines an arbitrary colour-preserving monotone induced subgraph isomorphism. The proof of Lemma 3.3 shows that $\mathfrak{d} \leq \mathfrak{a}^h$ holds. We can therefore say that the output index tuple defines the "maximum" or "rightmost" colour-preserving monotone induced subgraph isomorphism. This is a useful observation, useful when asking for isomorphisms of very special properties.

Another interesting observation about OIS concerns the chosen conflicting index pair i, j. The algorithm chooses an arbitrary such index pair among all possible conflicting index pairs, and the result of the algorithm, acceptance or rejection as well as the output index tuple, is not influenced by the actual choice. This is noteworthy for theoretical as well as implemention aspects.

Finally, consider the related subgraph isomorphism problem: given two graphs G and H, decide whether H is isomorphic to a (not necessarily induced) subgraph G' of G. We can modify OIS to decide this problem: conflicting index pairs are those i, j for which $y_i y_j \in E(H)$ and $x_{a_i} x_{a_j} \notin E(G)$. It is an easy exercise to modify the proofs of this section to show the following

result: there is a polynomial-time algorithm that decides on input $(G, f_G, \sigma;\ H, f_H, \tau)$ whether H is colour-preserving (σ,τ)-isomorphic to a subgraph of G, assuming that (G, f_G, σ) satisfies the right umbrella condition. If G is an interval graph, f_G is a 1-colouring for G and σ is an interval ordering for G, this is the case. Neglecting the (σ,τ)-monotonicity, the resulting Subgraph Isomorphism problem is NP-complete already on connected proper interval graphs [14].

4 ISI on proper interval graphs

Proper interval graphs are cocomparability graphs of special properties, and they have a characterisation through vertex orderings. Let G be a graph. A vertex ordering σ for G is called a *proper interval ordering* if for every vertex triple u, v, w of G with $u \prec_\sigma v \prec_\sigma w$, $uw \in E(G)$ implies $uv \in E(G)$ and $vw \in E(G)$. A graph is a *proper interval graph* if it has a proper interval ordering [15]. We only consider proper interval graphs through this vertex ordering characterisation. Observe for every vertex ordering σ for G: σ is a proper interval ordering for G if and only if the reverse of σ is a proper interval ordering for G. Also observe that every proper interval ordering is a cocomparability ordering. It can be decided in linear time whether a given graph is a proper interval graph, and if so, a proper interval ordering can be generated in linear time [15].

First, we prove that proper interval orderings are unique up to reversal and up to sets of vertices with the same neighbourhoods. To be more precise, let G be a connected proper interval graph and let σ and τ be proper interval orderings for G. Assume that $\sigma = \langle x_1, \ldots, x_n\rangle$ and $\tau = \langle y_1, \ldots, y_n\rangle$. We say that σ and τ are *strongly neighbourhood-equivalent* if $N_G[x_i] = N_G[y_i]$ for every $1 \leq i \leq n$. We will show that σ is strongly neighbourhood-equivalent to τ or to the reverse of τ. In order to prove this statement as Proposition 4.2 below, it suffices to prove the next lemma, whose technical statements will be referred to separately later in the paper. The first statement of Lemma 4.1 shows that if the leftmost vertices of σ and τ have the same closed neighbourhood then σ and τ are strongly neighbourhood-equivalent. The second statement of the lemma shows that the closed neighbourhoods of the first vertices of σ and τ are equal or the closed neighbourhoods of the first vertex of σ and the last vertex of τ are equal. Although Lemma 4.1 has been used implicitly in several previous works [4, 6, 12, 15], it has not been stated as a result on its own and proved separately before.

Lemma 4.1. *Let G be a connected proper interval graph. Let $\sigma = \langle x_1, \ldots, x_n\rangle$ and $\tau = \langle y_1, \ldots, y_n\rangle$ be proper interval orderings for G.*

1) *If $N_G[x_1] = N_G[y_1]$ then σ and τ are strongly neighbourhood-equivalent.*

2) *If $x_1 \prec_\tau x_n$ then $N_G[x_1] = N_G[y_1]$, and*
   if $x_n \prec_\tau x_1$ then $N_G[x_1] = N_G[y_n]$.

Proof. Since the two statements trivially hold for the case $n = 1$, we assume $n \geq 2$. The following properties of proper interval orderings will be important for the proof. Consider σ. Since G is connected, $x_i x_{i+1} \in E(G)$ for every $1 \leq i < n$. It directly follows that $(x_1, \ldots, x_n)$ is an x_1, x_n-path of G. Particularly note that $x_2 \in N_G[x_1]$. Furthermore, for every vertex triple u, v, w

of G with $u = x_1$ and $u \prec_\sigma v \preccurlyeq_\sigma w$ and $w \in N_G(u)$, it holds that $N_G[u] \subseteq N_G[v] \subseteq N_G[w]$. Since τ is also a proper interval ordering for G, the same properties analogously hold for τ.

Proof of 1)
We assume $N_G[x_1] = N_G[y_1]$. If $x_2 = y_2$ then $N_G[x_2] = N_G[y_2]$, if $x_2 = y_1$ then $y_1 \prec_\tau y_2 \preccurlyeq_\tau x_1$, and thus, $N_G[x_1] \subseteq N_G[x_2] = N_G[y_1] \subseteq N_G[y_2] \subseteq N_G[x_1]$, if $y_2 = x_1$ then, analogously, $N_G[y_2] = N_G[x_2]$, and if $x_1 \neq y_2$ and $x_2 \neq y_2$ and $x_2 \neq y_1$ then $x_1 \prec_\sigma x_2 \prec_\sigma y_2$ and $y_1 \prec_\tau y_2 \prec_\tau x_2$ and $N_G[x_2] \subseteq N_G[y_2]$ and $N_G[y_2] \subseteq N_G[x_2]$. Thus, $N_G[x_2] = N_G[y_2]$ in all cases.

Let j be such that $y_j = x_1$; note that $N_G[y_1] = N_G[x_1] = N_G[y_j]$. If $j = 1$ then $x_1 = y_1$, and $\langle x_2, \ldots, x_n\rangle$ and $\langle y_2, \ldots, y_n\rangle$ are proper interval orderings for $G - x_1$. If $j \geq 2$ then $\langle x_2, \ldots, x_n\rangle$ and $\langle y_2, \ldots, y_{j-1}, y_1, y_{j+1}, \ldots, y_n\rangle$ are proper interval orderings for $G - x_1$. In both cases, we conclude the claim by induction. Note that $G - x_1$ is indeed a connected proper interval graph.

Proof of 2)
If $x_1 = y_1$ then the claim trivially holds, if $x_1x_n \in E(G)$ then G is a complete graph, and the claim holds. So, as the remaining case, assume that $x_1 \neq y_1$ and $x_1x_n \notin E(G)$. Assume $x_1 \prec_\tau x_n$. Let p be such that $x_p = y_1$. Recall that $(x_p, \ldots, x_n)$ is an x_p, x_n-path of G. For a contradiction, suppose that $x_1y_1 \notin E(G)$. Then, $(x_p, \ldots, x_n)$ does not contain any vertex from $N_G[x_1]$, since $N_G[x_1] \subseteq \{x_1, \ldots, x_{p-1}\}$. We consider τ. Let $P = (u_0, \ldots, u_r)$ be an arbitrary y_1, x_n-path of G. Since $y_1 \prec_\tau x_1 \prec_\tau x_n$ and $u_0 = y_1$ and $u_r = x_n$, there is an index i with $0 \leq i < r$ such that $u_i \preccurlyeq_\tau x_1 \prec_\tau u_{i+1}$. Thus, x_1 is a vertex on P or P contains a neighbour of x_1, which follows from the definition of proper interval orderings, so that P contains a vertex from $N_G[x_1]$. By the choice of P as an arbitrary y_1, x_n-path of G, it follows that every such path of G contains a vertex from $N_G[x_1]$, contradicting the initial assumption. Thus, $x_1y_1 \in E(G)$. The results from the beginning of the proof then show that $N_G[x_1] \subseteq N_G[y_1]$ and $N_G[y_1] \subseteq N_G[x_1]$, i.e., $N_G[x_1] = N_G[y_1]$.

The remaining proof for the situation of $x_n \prec_\tau x_1$ follows from applying the first situation to σ and the reverse of τ. ∎

Lemma 4.1 readily implies the following.

Proposition 4.2. *Let G be a connected proper interval graph with proper interval orderings σ and τ. Then, σ is strongly neighbourhood-equivalent to τ or to the reverse of τ.*

We now prove another implication of Lemma 4.1, that will play a crucial role in the proof of the main result of this section.

Lemma 4.3. *Let G and H be proper interval graphs where H is connected. Let σ and τ be proper interval orderings for respectively G and H, and let τ^R be the reverse of τ. Then, H is isomorphic to an induced subgraph of G if and only if there is $\omega \in \{\tau, \tau^R\}$ such that H is (σ, ω)-isomorphic to an induced subgraph of G.*

Proof. If H is a graph on a single vertex, the claim trivially holds. We therefore assume that H has at least two vertices. Clearly, if H is (σ, τ)- or (σ, τ^R)-isomorphic to an induced subgraph of G then H is isomorphic to an induced subgraph of G.

For the converse, assume that G has an induced subgraph G' such that H is isomorphic to G'; let φ be an isomorphism from H to G'. Observe that G' is connected. Let σ' be the

restriction of σ to the vertices of G'. Observe that σ' is a proper interval ordering for G'. Assume that $\sigma' = \langle x'_1, \ldots, x'_r \rangle$ and $\tau = \langle y_1, \ldots, y_r \rangle$. Recall that $\tau^R = \langle y_r, \ldots, y_1 \rangle$ and $x'_1 \prec_\sigma \cdots \prec_\sigma x'_r$. Let $\varphi' : V(H) \to V(G')$ be the mapping where y_i is mapped to x'_i and let $\varphi'' : V(H) \to V(G')$ be the mapping where y_i is mapped to x'_{r+1-i}. We show that one of the two mappings is an isomorphism from H to G', which proves the claimed result.

First, assume that $\varphi(y_1) \prec_{\sigma'} \varphi(y_r)$. We show that φ' is an isomorphism from H to G', by showing for every vertex y of H that $N_{G'}[\varphi'(y)] = N_{G'}[\varphi(y)]$. Let $\sigma^* =_{\text{def}} \langle \varphi(y_1), \ldots, \varphi(y_r) \rangle$. Observe that σ^* is a proper interval ordering for G', and $\varphi(y_1) \prec_{\sigma^*} \varphi(y_r)$. Recall that σ' is also a proper interval ordering for G', and that $\varphi(y_1) \prec_{\sigma'} \varphi(y_r)$. Hence, we can apply Lemma 4.1 to find that σ^* and σ' are strongly neighbourhood-equivalent. Since $x'_i = \varphi'(y_i)$ for every $i \in \{1, \ldots, r\}$, we obtain the desired result. If $\varphi(y_r) \prec_{\sigma'} \varphi(y_1)$ then an analogous proof shows that φ'' is an isomorphism from H to G', where Lemma 4.1 is applied to $\langle \varphi(y_r), \ldots, \varphi(y_1) \rangle$ and σ'. ∎

We are now ready to give the main result of this section.

Theorem 4.4. *Given a proper interval graph G and a connected graph H, it can be decided in polynomial time whether G has an induced subgraph that is isomorphic to H.*

Proof. We describe such an algorithm. First, assume that G and H are connected proper interval graphs. Let σ and τ be proper interval orderings for respectively G and H. Let f_G and f_H be 1-colourings for respectively G and H. Let τ^R be the reverse of τ. We run Algorithm OIS on $(G, f_G, \sigma;\ H, f_H, \tau)$ and on $(G, f_G, \sigma;\ H, f_H, \tau^R)$ and accept if and only if OIS accepts on (at least) one of the two inputs. If H is not a proper interval graph, reject, and if G is not connected, apply the above procedure to every connected component of G.

Let us prove the correctness of the algorithm. As proper interval graphs are hereditary, we can correctly reject if H is not a proper interval graph. By Lemma 4.3, H is isomorphic to an induced subgraph of G if and only if H is (σ, τ)- or (σ, τ^R)-isomorphic to an induced subgraph of G. Proper interval orderings together with any colouring satisfy the left and right umbrella conditions, in particular, for the 1-colourings used in the algorithm. It follows from Corollary 3.4 that our algorithm will accept if and only if H is isomorphic to an induced subgraph of G.

For the running time, recall that it can be checked in linear time whether H is a proper interval graph, and a proper interval ordering for H can be computed in linear time [15]. Algorithm OIS has a polynomial running time and is applied two times for each connected component of G. Thus, the total running time of our algorithm is polynomial. ∎

5 ISI on bipartite permutation graphs

Bipartite permutation graphs are those graphs that are both bipartite graphs and permutation graphs. Like proper interval graphs, permutation graphs are cocomparability graphs and they admit a characterisation through vertex orderings. Fixing two horizontal lines (A and B in Figure 4), a *permutation diagram* is a set of line segments that connect points on the two horizontal lines and that share no endpoints. A graph G is called a *permutation graph* if it has a permutation diagram such that each vertex of G is associated with a line segment of the permutation diagram, and two vertices of G are adjacent if and only if the associated line segments intersect.

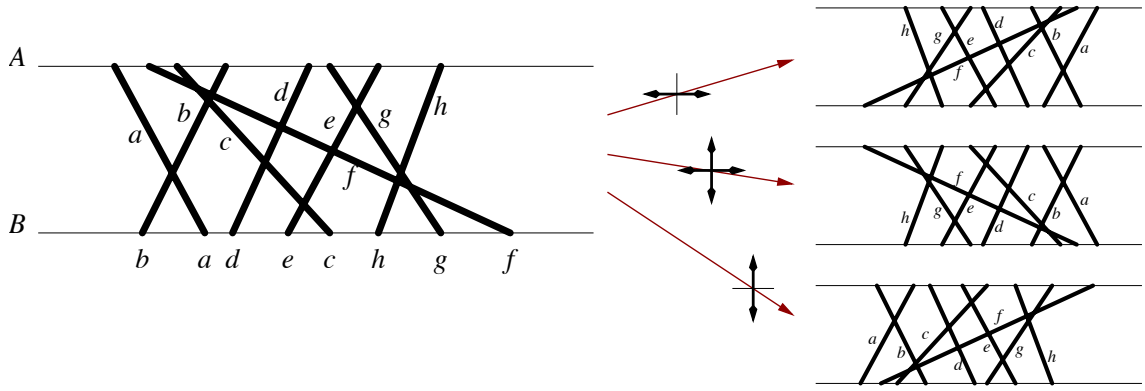

Figure 4: To the left, a permutation diagram, that represents a graph on eight vertices. To the right, from top to bottom, the three permutation diagrams that can be obtained from the left diagram by applying a horizontal flip, a horizontal and a vertical flip, and a vertical flip, respectively.

A vertex ordering that is both a cocomparability ordering and a comparability ordering is called a *permutation ordering*; recall the definitions of these vertex orderings from Section 2. Every permutation diagram defines a vertex ordering for the corresponding graph, where the ordering is obtained by reading the vertices of the line segments in their endpoint order on the lower horizontal line (B). It is not difficult to verify that such a vertex ordering is in fact a permutation ordering, and each permutation ordering defines a permutation diagram. Hence, a graph is a permutation graph if and only if it has a permutation ordering. Permutation orderings for permutation graphs can be computed in linear time [18].

Permutation diagrams are "flip invariant", meaning that flipping a permutation diagram horizontally or vertically results in a permutation diagram for the same graph. The horizontal and vertical flip operations are illustrated in Figure 4, together with all possible permutation diagrams that can be obtained from a given diagram by applying these operations. Given the permutation diagram on the left-hand side of Figure 4, the top diagram on the right-hand side is obtained by applying the horizontal flip operation, the bottom diagram on the right-hand side is obtained by applying the vertical flip operation, and the middle diagram on the right-hand side is obtained by applying the horizontal and the vertical flip operations. We call the three diagrams on the right-hand side of Figure 4 the *flip variants* of the diagram on the left. Every permutation diagram together with its three flip variants defines four permutation orderings for a permutation graph; for example, the permutation diagrams in Figure 4 define the following four permutation orderings: $\langle b, a, d, e, c, h, g, f\rangle$, $\langle f, g, h, c, e, d, a, b\rangle$, $\langle h, e, g, d, b, c, f, a\rangle$ and $\langle a, f, c, b, d, g, e, h\rangle$.

Let G be a connected permutation graph with permutation diagram $\mathcal{D}$, and let σ be the permutation ordering for G defined by $\mathcal{D}$. By $\mathrm{perm}(\sigma)$, we denote the set of the permutation orderings for G that are defined by $\mathcal{D}$ and its flipping variants. Note that $\mathrm{perm}(\sigma)$ contains σ and it contains at most four permutation orderings for G. For the example of Figure 4, we

obtain:

$$\begin{aligned} & \mathrm{perm}(\langle b,a,d,e,c,h,g,f\rangle) \;=\; \mathrm{perm}(\langle f,g,h,c,e,d,a,b\rangle) \\ = \;& \mathrm{perm}(\langle h,e,g,d,b,c,f,a\rangle) \;=\; \mathrm{perm}(\langle a,f,c,b,d,g,e,h\rangle) \\ = \;& \Big\{\langle b,a,d,e,c,h,g,f\rangle, \langle f,g,h,c,e,d,a,b\rangle, \langle h,e,g,d,b,c,f,a\rangle, \langle a,f,c,b,d,g,e,h\rangle\Big\}. \end{aligned}$$

A graph is a *bipartite permutation graph* if it is both a permutation graph and a bipartite graph. Since bipartite graphs have no cycles of odd length and cocomparability graphs have no induced cycles of length more than 4 [10], it follows that a cocomparability graph is bipartite if and only if it has no triangles, i.e., if and only if it has no cycles of length 3. This can be translated into a vertex ordering characterisation of bipartite permutation graphs.

Lemma 5.1. *Let G be a graph and let σ be a vertex ordering for G. Then, G is a bipartite permutation graph with permutation ordering σ if and only if σ is a cocomparability ordering for G satisfying for every vertex triple u,v,w of G with $u \prec_\sigma v \prec_\sigma w$: $uv \notin E(G)$ or $vw \notin E(G)$.*

Proof. Assume that G is a bipartite permutation graph and σ is a permutation ordering for G. Since σ is a cocomparability ordering for G, it remains to verify the second condition. Let u,v,w be a vertex triple of G with $u \prec_\sigma v \prec_\sigma w$. If $uv, vw \in E(G)$ then $uw \in E(G)$ by the definition of comparability orderings, and thus, G has a triangle, a contradiction to being bipartite. Hence, $uv \notin E(G)$ or $vw \notin E(G)$.

For the converse, let σ be a cocomparability ordering for G satisfying the vertex triple condition. It directly follows that σ is a comparability ordering for G, and thus, G is a permutation graph with permutation ordering σ. Suppose for a contradiction that G is not bipartite. Then, G contains a triangle, i.e., G has a vertex triple u,v,w with $u \prec_\sigma v \prec_\sigma w$ and $uv, vw, uw \in E(G)$. This violates the vertex triple condition, a contradiction. ∎

In this section, we mainly view bipartite permutation graphs through their permutation orderings. These permutation orderings are fully characterised by the result of Lemma 5.1. We will show that permutation orderings for bipartite permutation graphs are unique up to the aforementioned flip operations and up to sets of vertices with the same neighbourhoods. Let G be a connected bipartite permutation graph and let $\sigma = \langle x_1, \ldots, x_n\rangle$ and $\tau = \langle y_1, \ldots, y_n\rangle$ be two permutation orderings for G. We say that σ and τ are *neighbourhood-equivalent* if $N_G(x_i) = N_G(y_i)$ for every $1 \le i \le n$. Observe that, in contrast to the definition of strongly neighbourhood-equivalent in Section 4, we are now comparing open neighbourhoods.

In Proposition 5.4 below, we will show that τ, and thus every permutation ordering for G, is neighbourhood-equivalent to some permutation ordering in $\mathrm{perm}(\sigma)$. In order to prove this, we need to prove a technical lemma that strongly resembles Lemma 4.1 for proper interval orderings. The first statement of Lemma 5.3 below shows that if the leftmost vertices of σ and τ have the same open neighbourhood then the two orderings are neighbourhood-equivalent. In the second and third statement of Lemma 5.3, we identify three vertices in G such that the open neighbourhood of the leftmost vertex of τ is equal to the open neighbourhood of one of these three vertices. Before we formally state and prove Lemma 5.3, let us point out that this lemma is used implicitly in the correctness proof of a recognition algorithm for bipartite permutation

graphs due to Spinrad, Brandstädt and Stewart [20]. Our proof however uses an alternative approach, solely based on permutation orderings and the characterisation of Lemma 5.1.

In the proof of Lemma 5.3 below, we will frequently use the following notion.

Definition 5.2. *Let G be a graph and let σ be a vertex ordering for G. A triple (u, v, w) of vertices of G is* σ-conflicting *if $u \prec_\sigma v \prec_\sigma w$ and if at least one of the following conditions holds:*

(i) $uw \in E(G)$ and neither $uv \in E(G)$ nor $vw \in E(G)$;

(ii) $uv \in E(G)$ and $vw \in E(G)$.

Lemma 5.3. *Let G be a connected bipartite permutation graph on at least two vertices, and let $\sigma = \langle x_1, \ldots, x_n \rangle$ and $\tau = \langle y_1, \ldots, y_n \rangle$ be permutation orderings for G. Let z be the neighbour of x_1 such that there is no neighbour u of x_1 with $x_1 \prec_\sigma u \prec_\sigma z$, and let z' be the neighbour of x_n such that there is no neighbour v of x_n with $z' \prec_\sigma v \prec_\sigma x_n$.*

1) If $N_G(x_1) = N_G(y_1)$ then σ and τ are neighbourhood-equivalent.

2) If $x_1 \prec_\tau x_n$ and $x_1x_n \in E(G)$ then $N_G(y_1) = N_G(x_1)$ or $N_G(y_1) = N_G(z')$.

3) If $x_1 \prec_\tau x_n$ and $x_1x_n \notin E(G)$ then $N_G(y_1) = N_G(x_1)$ or $N_G(y_1) = N_G(z)$.

Proof. We can say that z is the closest neighbour of x_1 in σ and z' is the closest neighbour of x_n in σ. From the definition of a cocomparability ordering and Lemma 5.1, it follows that G contains neither a σ-conflicting triple nor a τ-conflicting triple. In particular, this means that for any vertex v of G, either all the neighbors of v appear to the left of v in the ordering σ (respectively τ), or they all appear to the right of v. We will often make use of these observations in the proof below. We will also use the following claim.

Claim A. Let u and v be two vertices of G. If $u, v \in N_G(x_n)$ and $u \prec_\sigma v \preccurlyeq_\sigma x_n$, then $N_G(v) \subseteq N_G(u)$. If $u, v \in N_G(x_1)$ and $x_1 \preccurlyeq_\sigma u \prec_\sigma v$, then $N_G(u) \subseteq N_G(v)$.

To prove Claim A, first suppose that $u, v \in N_G(x_n)$ and $u \prec_\sigma v \preccurlyeq_\sigma x_n$. Note that $uv \notin E(G)$, as otherwise (u, v, x_n) would be σ-conflicting. Let $w \in N_G(v) \setminus \{x_n\}$. Recall that the neighbors of v either all appear to the left or all appear to the right of v in σ. Since x_n is a neighbor of v and $v \prec_\sigma x_n$, we have $v \prec_\sigma w \prec_\sigma x_n$. Since (v, w, x_n) is not σ-conflicting, $wx_n \notin E(G)$. But then $u \in N_G(w)$, as otherwise (u, w, x_n) would be σ-conflicting. Hence $N_G(v) \subseteq N_G(u)$. The second statement of Claim A can be proved using symmetrical arguments.

Proof of 1)
We prove the first statement by induction on n. It is clear that the statement holds if $n = 2$, so we assume $n \geq 3$. We assume $N_G(x_1) = N_G(y_1)$ and start by showing that this implies $N_G(x_2) = N_G(y_2)$. Since the latter equality is trivially true if $x_2 = y_2$, we assume $x_2 \neq y_2$. We distinguish between two cases.

As the first case, we assume that $x_1x_2 \in E(G)$ or $y_1y_2 \in E(G)$. Without loss of generality, assume $x_1x_2 \in E(G)$. Note that $N_G(x_2) = \{x_1\}$, since if x_2 had any neighbor $t \neq x_1$, then the triple (x_1, x_2, t) would be σ-conflicting. The assumption $N_G(x_1) = N_G(y_1)$ implies that $x_1 = y_1$. Moreover, since $y_2 \neq x_2$, we obtain $y_1 \prec_\tau y_2 \prec_\tau x_2$. The fact that $y_1x_2 \in E(G)$ and $y_2x_2 \notin E(G)$ implies that $y_1y_2 \in E(G)$, as otherwise the triple (y_1, y_2, x_2) would be τ-conflicting. This in turn

implies that y_2 has no neighbor $t \neq y_1$, as otherwise the triple (y_1, y_2, t) would be τ-conflicting. Hence $N_G(y_2) = \{y_1\}$, which yields the desired equality $N_G(y_2) = \{y_1\} = \{x_1\} = N_G(x_2)$.

As the second case, we assume $x_1x_2 \notin E(G)$ and $y_1y_2 \notin E(G)$. Since z is a neighbour of x_1 and therefore of y_1, it holds that $x_2 \neq z$ and $y_2 \neq z$, and consequently $x_1 \prec_\sigma x_2 \prec_\sigma z$ and $y_1 \prec_\tau y_2 \prec_\tau z$. Since σ and τ are cocomparability orderings, z is adjacent to x_2 and y_2. We locate y_2 in σ and x_2 in τ. Note that $y_2 \prec_\sigma z$, as otherwise the triple (x_1, z, y_2) would be σ-conflicting. Similarly, we must have $x_2 \prec_\sigma z$ as otherwise (y_1, z, x_2) would be σ-conflicting. Thus, $x_1 \prec_\sigma x_2 \prec_\sigma y_2 \prec_\sigma z$ and $y_1 \prec_\tau y_2 \prec_\tau x_2 \prec_\tau z$. This implies $x_2y_2 \notin E(G)$, as otherwise the triples (x_2, y_2, z) and (y_2, x_2, z) would be σ-conflicting and τ-conflicting, respectively. In order to show that $N_G(x_2) \subseteq N_G(y_2)$, let u be a neighbour of x_2 in G. Since z is a neighbor of x_2 and $x_2 \prec_\sigma z$, all neighors of x_2 appear to the right of x_2 in σ. In particular, $x_2 \prec_\sigma u$. We claim that $z \prec_\sigma u$. For a contradiction, suppose $x_1 \prec_\sigma x_2 \prec_\sigma u \prec_\sigma z$. Note that $x_1u \notin E(G)$ by the definition of z, and $uz \notin E(G)$ as otherwise (x_2, u, z) would be a σ-conflicting triple. But the existence of the edge x_1z implies that (x_1, y_2, z) is a σ-conflicting triple, yielding the desired contradiction. Hence $x_2 \prec_\sigma y_2 \prec_\sigma z \prec_\sigma u$. Since $x_2u \in E(G)$, $x_2y_2 \notin E(G)$ and (x_2, y_2, u) is not σ-conflicting, u is a neighbour of y_2, which implies that $N_G(x_2) \subseteq N_G(y_2)$. Analogously, every neighbor of y_2 must be adjacent to x_2, implying that $N_G(y_2) \subseteq N_G(x_2)$, and hence $N_G(x_2) = N_G(y_2)$.

To complete the proof, let us first consider the case where $x_1x_2 \in E(G)$. Recall that $N_G(x_2) = \{x_1\}$ in this case. Let $G' = G - x_2$, and observe that G' is connected. Let σ' and τ' be the two vertex orderings for G' obtained from σ and τ, respectively, by removing vertex x_2. It is clear that σ' and τ' are permutation orderings for G'. Since $N_G(x_2) = N_G(y_2)$ and hence $N_{G'}(x_2) = N_{G'}(y_2)$, the orderings σ' and τ' are neighborhood-equivalent by induction, which implies that σ and τ are neighborhood-equivalent as well. The case where $y_1y_2 \in E(G)$ can be proved analogously. Finally, consider the case where $x_1x_2 \notin E(G)$ and $y_1y_2 \notin E(G)$. Recall that $N_G(x_1) \subseteq N_G(x_2)$ in this case. Consequently, $G - x_1$ is connected. Hence we can use $G - x_1$ instead of $G - x_2$ to conclude that σ and τ are neighborhood-equivalent also in this case.

Proof of 2)
We assume $x_1 \prec_\tau x_n$ and $x_1x_n \in E(G)$ and $N_G(y_1) \neq N_G(x_1)$, and we are going to show $N_G(y_1) = N_G(z')$. Observe that $y_1x_1 \notin E(G)$, as otherwise the triple (y_1, x_1, x_n) would be τ-conflicting. This, together with the fact that (x_1, y_1, x_n) is not σ-conflicting, implies that $y_1 \in N_G(x_n)$. Hence $N_G(z') \subseteq N_G(y_1)$ as an immediate result of Claim A and the definition of z'.

It remains to prove that $N_G(y_1) \subseteq N_G(z')$. This is trivially true if $y_1 = z'$, so we assume $y_1 \neq z'$. Since x_1 and y_1 are both neighbors of x_n and $x_1 \prec_\sigma y_1 \prec_\sigma x_n$, it follows from Claim A that $N_G(y_1) \subseteq N_G(x_1)$. This, together with the initial assumption $N_G(x_1) \neq N_G(y_1)$, implies that $N_G(y_1) \subset N_G(x_1)$. Let $a \in N_G(x_1) \setminus N_G(y_1)$. Observe that $x_1x_n \in E(G)$ and $ax_1 \in E(G)$ implies $ax_n \notin E(G)$, as otherwise (x_1, a, x_n) would be a σ-conflicting triple. Also note that $a \prec_\sigma y_1$, as otherwise (y_1, a, x_n) would be a σ-conflicting triple. Since $y_1 \prec_\sigma z'$ by the definition of z', we have $a \prec_\sigma z'$. The fact that $x_1a \in E(G)$ and (x_1, a, z') is not σ-conflicting implies that $az' \notin E(G)$. Similarly, $x_1z' \notin E(G)$, as otherwise (x_1, z', x_n) would be σ-conflicting.

We now determine the order of the vertices y_1, x_1, x_n, a, z' in τ, using the adjacencies and non-adjacencies established above and the fact that τ is a cocomparability ordering. Recall

that $y_1a \notin E(G)$, $ax_n \notin E(G)$, and $y_1x_n \in E(G)$. This implies that $y_1 \prec_\tau x_1 \prec_\tau x_n \prec_\tau a$. Similarly, $x_nz' \in E(G)$ and $x_na \notin E(G)$ and $az' \notin E(G)$ implies $z' \prec_\tau a$. Finally, $x_1a \in E(G)$ and $x_1z' \notin E(G)$ and $z'a \notin E(G)$ and $z' \prec_\tau a$ implies $z' \prec_\tau x_1$. Summarizing, we have $y_1 \prec_\tau z' \prec_\tau x_1 \prec_\tau x_n \prec_\tau a$.

To show that $N_G(y_1) \subseteq N_G(z')$, let b be a neighbor of y_1. Since x_1 and y_1 are neighbors of x_n and $x_1 \prec_\sigma y_1 \prec_\sigma x_n$, we have that $N_G(y_1) \subseteq N_G(x_1)$ as a result of Claim A. In particular, we have $bx_1 \in E(G)$. We observe that $x_1 \prec_\tau b$, as otherwise (y_1, b, x_1) would be τ-conflicting, and that $y_1z' \notin E(G)$, as otherwise (y_1, z', x_n) would be τ-conflicting. Hence we have $y_1 \prec_\tau z' \prec_\tau b$. Since $y_1b \in E(G)$ and (y_1, z', b) is not τ-conflicting, implies that $z'b \in E(G)$. We conclude that $N_G(y_1) \subseteq N_G(z')$.

Proof of 3)
We assume $x_1 \prec_\tau x_n$ and $x_1x_n \notin E(G)$ and $x_1 \neq y_1$. We distinguish between the two cases $x_1y_1 \in E(G)$ and $x_1y_1 \notin E(G)$. We are going to show $N_G(y_1) = N_G(z)$ in the former case and $N_G(y_1) = N_G(x_1)$ in the latter case.

For the first case, let $x_1y_1 \in E(G)$. Since $N_G(y_1) = N_G(z)$ trivially holds if $y_1 = z$, we assume that $y_1 \neq z$. It follows from the definition of z and Claim A that $N_G(z) \subseteq N_G(y_1)$. It remains to show that $N_G(y_1) \subseteq N_G(z)$. Suppose for a contradiction that there exists a vertex $a \in N_G(y_1) \setminus N_G(z)$. The properties of σ imply the following order: $x_1 \prec_\sigma z \prec_\sigma a \prec_\sigma y_1$. We use this to prove the following claim.

Claim B. There is a y_1, x_n-path P of G such that $z \prec_\sigma v$ for every vertex v of P.
Let $Q = (u_0, \ldots, u_k)$ be a y_1, x_n-path of G; note that such a path exists since G is connected. Let $i \in \{0, 1, \ldots, k-1\}$ be the largest index such that $u_i \preccurlyeq_\sigma y_1$. Note that $u_i \neq y_1$, as otherwise (x_1, y_1, u_{i+1}) would be a σ-conflicting triple, so $u_i \prec_\sigma y_1$. Similarly, since (u_i, y_1, u_{i+1}) is not σ-conflicting, it holds that $y_1u_{i+1} \notin E(G)$. The edge u_iu_{i+1} and the fact that σ is a cocomparability ordering implies that y_1 is adjacent to u_i. Hence, if $z \prec_\sigma u_i$, then we can take P to be the path $(y_1, u_i, u_{i+1}, \ldots, u_k)$. Suppose $u_i \prec_\sigma z$, which implies that $u_i \prec_\sigma a \prec_\sigma y_1 \prec_\sigma u_{i+1}$. We consider the triple (u_i, a, u_{i+1}). Since this triple is not σ-conflicting, we must have $u_ia \in E(G)$ or $au_{i+1} \in E(G)$. The former is not possible, as then (u_i, a, y_1) would be a σ-conflicting triple. Hence $au_{i+1} \in E(G)$, so we can take P to be the path $(y_1, a, u_{i+1}, \ldots, u_k)$.

Let P be a y_1, x_n-path of G whose vertices all appear to the right of z in the ordering σ; the existence of P is guaranteed by Claim B. Since x_1 is a neighbor of z and $x_1 \prec_\sigma z$, all the neighbors of z appear to the left of z in σ. As a result, the path P contains no vertex of $N_G[z]$. Now consider the ordering τ. By assumption, we have $y_1 \prec_\tau x_1 \prec_\tau x_n$. We also have $y_1 \prec_\tau z \prec_\tau x_1$, as $x_1 \prec_\tau z$ would imply that (y_1, x_1, z) is τ-conflicting. By the properties of the cocomparability ordering τ, every path from y_1 to x_n must contain a vertex from $N_G[z]$. However, the path P does not contain any vertex from $N_G[z]$, yielding the desired contradiction. We conclude that $N_G(y_1) \subseteq N_G(z)$, and hence $N_G(y_1) = N_G(z)$. This completes the proof of the first case.

For the second case, let $x_1y_1 \notin E(G)$. We will show that $N_G(y_1) = N_G(x_1)$ in this case. We first show that all the neighbors of x_1 appear to the right of x_1 in τ. Suppose for a contradiction that some neighbor p of x_1 appears to the left of x_1 in τ. Recall that this implies that all neighbors of x_1 appear to the left of x_1 in τ. Then we must have $q \prec_\tau x_1$ for every $q \in N_G(y_1)$,

as otherwise (y_1, x_1, q) would be τ-conflicting. In other words, all neighbors of y_1 appear to the left of x_1 in τ. Let $P = (u_0, \ldots, u_k)$ be an x_1, x_n-path of G, and let $i \in \{0, 1, \ldots, k-1\}$ be the largest index such that $u_i \preccurlyeq_\tau x_1 \prec_\tau u_{i+1}$. Note that $u_i \prec_\tau x_1$, as otherwise (p, u_i, u_{i+1}) would be τ-conflicting. Moreover, $x_1 u_{i+1} \notin E(G)$ due to the fact that (p, x_1, u_{i+1}) is not τ-conflicting. This implies that $u_i x_1 \in E(G)$, as otherwise (u_i, x_1, u_{i+1}) would be τ-conflicting. Let Q be the path $(x_1, u_i, u_{i+1}, \ldots, u_k)$. Since $x_1 \prec_\sigma y_1 \prec_\sigma x_n$ and σ is a cocomparability ordering for G, every x_1, x_n-path of G, and path Q in particular, contains a vertex from $N_G[y_1]$. Recall that all the neighbors of y_1 appear to the left of x_1 in τ, implying that $\{x_1, u_{i+1}, \ldots, u_k\} \cap N_G(y_1) = \emptyset$. Hence, it must hold that $y_1 u_i \in E(G)$. But then the triple (y_1, u_i, u_{i+1}) is τ-conflicting, yielding the desired contradiction. We conclude all the neighbors of x_1 appear to the right of x_1 in τ.

Now let a and b be the "rightmost" neighbours of respectively x_1 and y_1 in τ, i.e., a and b are the vertices of G with $a \in N_G(x_1)$ and $b \in N_G(y_1)$ and such that $u \preccurlyeq_\tau a$ for every $u \in N_G(x_1)$ and $v \preccurlyeq_\tau b$ for every $v \in N_G(y_1)$. We claim that $a = b$. First, suppose for a contradiction that $a \prec_\tau b$. By the definition of a, this means that $x_1 b \notin E(G)$. However, since $y_1 x_1 \notin E(G)$ by assumption, the triple (y_1, x_1, b) is τ-conflicting. This contradiction shows that $b \preccurlyeq_\tau a$. Now suppose, again for a contradiction, that $b \prec_\tau a$. Then $y_1 a \notin E(G)$ by the definition of b. If $x_1 \prec_\tau x_n \prec_\tau a$, then $x_n a \in E(G)$ as otherwise (x_1, x_n, a) would be τ-conflicting, but then (x_1, a, x_n) is a σ-conflicting triple. Hence, we must have $x_1 \prec_\tau a \prec_\tau x_n$.

Let $Q = (w_0, \ldots, w_l)$ be an x_1, x_n-path of G. Let $i \in \{0, 1, \ldots, l-1\}$ be the largest index such that $w_i \preccurlyeq_\tau a \prec_\tau w_{i+1}$. Note that $w_i \neq a$, as otherwise (x_1, w_i, w_{i+1}) would be a τ-conflicting triple. Analogous to previous considerations, we find that $w_i \prec_\tau a \prec_\tau w_{i+1}$ and $w_i a \in E(G)$. Hence $R = (x_1, a, w_i, w_{i+1}, \ldots, w_l)$ is an x_1, x_n-path of G. Recall that every x_1, x_n-path, and path R in particular, contains a vertex from $N_G[y_1]$. Note that $y_1 x_1 \notin E(G)$ by assumption. Also note that the rightmost neighbor b of y_1 satisfies $b \prec_\tau a$, and that $a \prec_\tau w_j$ for every $j \in \{i+1, \ldots, l\}$ by the definition of index i. Hence we must have $y_1 w_i \in E(G)$. But then (y_1, a, w_i) is a τ-conflicting triple. This contradiction implies that $a = b$.

In order to show that $N_G(x_1) \subseteq N_G(y_1)$, let w be a neighbour of x_1. If $w = a$, then $w \in N_G(y_1)$. Suppose $w \neq a$. Then $y_1 \prec_\tau x_1 \prec_\tau w \prec_\tau a$. Since (x_1, w, a) is not τ-conflicting and $x_1 w, x_1 a \in E(G)$, we must have $wa \notin E(G)$. Then the fact that (y_1, w, a) is not τ-conflicting implies that $y_1 w \in E(G)$. Analogously, by exchanging the roles of σ and τ and x_1 and y_1, we can show that $N_G(y_1) \subseteq N_G(x_1)$. We conclude that $N_G(x_1) = N_G(y_1)$. ∎

We use Lemma 5.3 to prove the following result.

Proposition 5.4. *Let G be a connected bipartite permutation graph with permutation orderings σ and τ. Then, there is $\omega \in \mathrm{perm}(\sigma)$ such that ω and τ are neighbourhood-equivalent.*

Proof. We can henceforth assume that G has at least two vertices. In particular, each vertex of G has a neighbour. Assume $\sigma = \langle x_1, \ldots, x_n \rangle$ and $\tau = \langle y_1, \ldots, y_n \rangle$. We apply the statements of Lemma 5.3 and the flipping of permutation diagrams from the beginning of the section to prove the result. Let $\mathcal{D}$ be a permutation diagram for G that defines the permutation ordering σ.

If $N_G(x_1) = N_G(y_1)$ then σ and τ are neighbourhood-equivalent due to the first statement of Lemma 5.3, and the result follows by recalling $\sigma \in \mathrm{perm}(\sigma)$.

Suppose $N_G(x_1) \neq N_G(y_1)$. Assume that x_1 and x_n are adjacent in G, i.e., $x_1 x_n \in E(G)$, and also assume $x_1 \prec_\tau x_n$, which means that x_1 and x_n appear in the same order in σ and τ.

We can apply the second statement of Lemma 5.3, and $N_G(y_1) = N_G(z')$ for z' the rightmost neighbour of x_n in σ. Let $\mathcal{D}'$ be the permutation diagram for G that is obtained from $\mathcal{D}$ by applying the vertical flip operation (see the bottom permutation diagram on the right-hand side of Figure 4), and let σ' be the permutation ordering defined by $\mathcal{D}'$. Note that $z' \prec_\sigma x_n$ and $x_n \prec_{\sigma'} z'$. Let a be the rightmost vertex in σ', and we are going to show that $a = z'$. Suppose for a contradiction that $a \neq z'$. Observe the following: $a \prec_\sigma x_n$, since x_n is the rightmost vertex in σ, and $x_n \prec_{\sigma'} z' \prec_{\sigma'} a$, since a is the rightmost vertex in σ'. It follows about a and x_n that a is a neighbour of x_n, and thus, $a \prec_\sigma z' \prec_\sigma x_n$ according to the choice of z' as being the rightmost neighbour of x_n. It follows that a and z' must also be adjacent, contradicting the vertex triple condition of Lemma 5.1. So, z' is the rightmost vertex in σ'.

Now, let $\mathcal{D}''$ be the permutation diagram obtained from $\mathcal{D}'$ by applying a horizontal flip, and let σ'' be the permutation ordering for G defined by $\mathcal{D}''$. Then, z' is the leftmost vertex of σ'', and since $N_G(y_1) = N_G(z')$, the first statement of Lemma 5.3 shows that τ and σ'' are neighbourhood-equivalent. Since $\sigma'' \in \mathrm{perm}(\sigma)$, the claim follows.

If $x_1x_n \in E(G)$ and $x_n \prec_\tau x_1$, we apply the above arguments to the reverse of τ, and if $x_1x_n \notin E(G)$, similar arguments also show the claimed result. ∎

The following lemma is the analogue of Lemma 4.3 for bipartite permutation graphs.

Lemma 5.5. *Let G and H be bipartite permutation graphs where H is connected. Let σ and τ be permutation orderings for respectively G and H. Then, H is isomorphic to an induced subgraph of G if and only if there is $\omega \in \mathrm{perm}(\tau)$ such that H is (σ, ω)-isomorphic to an induced subgraph of G.*

Proof. If H is (σ, ω)-isomorphic to an induced subgraph of G for some vertex ordering ω for H then H is isomorphic to an induced subgraph of G.

For the converse, we assume that H has at least two vertices and that H is isomorphic to an induced subgraph G' of G, via isomorphism φ. Observe that φ^{-1}, the inverse of φ, is an isomorphism from G' to H. Let $\sigma' = \langle x'_1, \ldots, x'_r \rangle$ be the restriction of σ to the vertices of G', which in particular means $x'_1 \prec_\sigma \cdots \prec_\sigma x'_r$. Observe that σ' is a permutation ordering for G'. Let $\tau' =_{\mathrm{def}} \langle \varphi^{-1}(x'_1), \ldots, \varphi^{-1}(x'_r) \rangle$. Since G' is isomorphic to H via isomorphism φ^{-1}, it holds that H is (σ, τ')-isomorphic to G' and that τ' is a permutation ordering for H. Due to Proposition 5.4, there is $\omega \in \mathrm{perm}(\tau)$ such that τ' and ω are neighbourhood-equivalent. We conclude that H is (σ, ω)-isomorphic to G' for some $\omega \in \mathrm{perm}(\tau)$. ∎

We are ready to prove the main result of this section.

Theorem 5.6. *Given a bipartite permutation graph G and a connected graph H, it can be decided in polynomial time whether G has an induced subgraph that is isomorphic to H.*

Proof. We describe such an algorithm, which is similar to the algorithm of Theorem 4.4. Assume that G and H are connected bipartite permutation graphs on at least two vertices. Let σ and τ be permutation orderings for respectively G and H. Let f_G and f_H be proper 2-colourings for respectively G and H; note that f_G and f_H exist due to the fact that G and H are bipartite graphs. It is well-known that every connected bipartite graph on at least two vertices has exactly two different proper 2-colourings, and that each of them can be obtained from the other by swapping the two colours. Let f'_H be the proper 2-colouring for H obtained from f_H by

swapping the two colours, i.e., by setting, for every vertex x of H, $f'_H(x) = 1$ if $f_H(x) = 2$ and $f'_H(x) = 2$ if $f_H(x) = 1$. Recall that $\mathrm{perm}(\tau)$ contains at most four permutation orderings for H. For every $\omega \in \mathrm{perm}(\tau)$ and $h \in \{f_H, f'_H\}$, we run Algorithm OIS on input $(G, f_G, \sigma;\ H, h, \omega)$ and accept if and only if OIS accepts on (at least) one of the at most eight inputs. If H is not a bipartite permutation graph, reject, and if G is not connected, apply the above procedure to every connected component of G.

Let us argue for the correctness of the algorithm. As bipartite permutation graphs are hereditary, we can correctly reject if H is not a bipartite permutation graph. By Lemma 5.5, H is isomorphic to an induced subgraph of G if and only if there is $\omega \in \mathrm{perm}(\tau)$ such that H is (σ, ω)-isomorphic to an induced subgraph of G. Recall that f_H and f'_H are the only two proper 2-colourings for H, and that $f_H(x) \neq f'_H(x)$ for every vertex x of H. Consequently, if H is isomorphic to an induced subgraph G' of G then H is colour-preserving isomorphic to G' for exactly one of the two colouring pairs (f_G, f_H) and (f_G, f'_H). From the properties of permutation orderings and the fact that f_G is a proper colouring for G, it follows that (G, f_G, σ) satisfies the left and right umbrella conditions. Thus, our algorithm accepts if and only if H is isomorphic to some induced subgraph of G.

For the running time, recall that recognising bipartite permutation graphs and computing a corresponding permutation ordering can be done in linear time [20]. The permutation orderings from $\mathrm{perm}(\tau)$ can be computed in linear time, by reading off the permutation diagram. A proper 2-colouring for a graph can be computed in linear time. So, the at most eight inputs for OIS can be generated in overall linear time, which gives a total polynomial running time. ∎

6 Fixed-parameter tractability and conclusion

We have seen that ISI is solvable in polynomial time if both G and H are proper interval graphs or if both G and H are bipartite permutation graphs, provided that H is connected. What happens when H is disconnected? We start with a simple observation.

Lemma 6.1. *Let H be a disconnected cocomparability graph with cocomparability ordering σ. Let C and D be two different connected components of H. Then, one of the following two cases applies:*

1) $x \prec_\sigma y$ for every vertex pair x, y with $x \in V(C)$ and $y \in V(D)$

2) $y \prec_\sigma x$ for every vertex pair x, y with $x \in V(C)$ and $y \in V(D)$.

Proof. Let a, x be a vertex pair of C and let b be a vertex of D, and assume that $a \prec_\sigma x$. If $a \prec_\sigma b \prec_\sigma x$ then b is adjacent to a vertex on every a, x-path of H as a consequence of the properties of cocomparability orderings. Such a path exists, since a and x belong to the same connected component of H. Since b does not belong to C, we conclude $b \prec_\sigma a \prec_\sigma x$ or $a \prec_\sigma x \prec_\sigma b$, which proves the lemma. ∎

We use the above lemma to prove the following result, which implies that ISI on proper interval graphs and bipartite permutation graphs is fixed-parameter tractable when parametrised by the number of connected components of H.

Theorem 6.2. *Given two graphs G and H such that G is a proper interval graph or a bipartite permutation graph on n vertices and H is a graph with k connected components, it can be decided in $k! \cdot n^{\mathcal{O}(1)}$ time whether H is isomorphic to an induced subgraph of G.*

Proof. We can assume that G and H are from the same graph class. For now, assume that both G and H are bipartite permutation graphs; we will briefly discuss the case when G and H are proper interval graphs at the end of the proof. Before we present the algorithm that decides whether or not H is isomorphic to an induced subgraph of G, we first make some useful structural observations.

Assume that H is isomorphic to an induced subgraph of G via isomorphism φ. Let $\sigma = \langle x_1, \ldots, x_n \rangle$ be a permutation ordering for G, and let f_G be a proper 2-colouring for G. Let $H_1, \ldots, H_k$ be the connected components of H. Each connected component of H is mapped by φ to an induced subgraph of some connected component of G, where different connected components of H may be mapped to the same connected component of G. To be more precise, there is a bijective mapping $\psi : \{1, \ldots, k\} \to \{1, \ldots, k\}$ such that for every vertex pair x, y of H and every index pair i, j with $1 \leq i, j \leq k$, if $x \in V(H_i)$ and $y \in V(H_j)$ and $i \neq j$ then $\varphi(x) \prec_\sigma \varphi(y)$ if and only if $\psi(i) < \psi(j)$. In other words, vertex ordering σ can be partitioned into k blocks $\sigma_1, \ldots, \sigma_k$ such that σ is obtained from appending the k blocks, i.e., $\sigma = \sigma_1 \circ \cdots \circ \sigma_k$, and the vertices from each connected component H_i are mapped to vertices in the block $\sigma_{\psi(i)}$.

Now, consider the connected component H_i of H with $\psi(i) = k$. Note that for any vertex x of H that does not belong to H_i, it holds that $\varphi(x) \prec_\sigma \varphi(y)$ for every vertex y of H_i. In other words, the images of the vertices of H_i appear to the right of the image of any other vertex of H. Let G' be the subgraph of G induced by the vertices of σ_k. Observe that G' contains the image of every vertex of H_i, but does not contain the image of any other vertex of H. Let τ_i be a permutation ordering for H_i, and let f_i and f_i' be two different proper 2-colourings of H_i. Using the exact same arguments as in the proof of Theorem 5.6, it holds that H_i is isomorphic to an induced subgraph of G if and only if there is a (σ, ω)-monotone isomorphism φ_i from H to an induced subgraph of G such that φ_i is colour-preserving for (G, f_G) and (H_i, f) for some $f \in \{f_i, f_i'\}$, i.e., if and only if Algorithm OIS outputs an index tuple $\mathfrak{a}^h = (a_1, \ldots, a_k)$ and accepts on input $(G, f_G, \sigma;\ H_i, f, \omega)$ for some $\omega \in \text{perm}(\tau_i)$ and $f \in \{f_i, f_i'\}$. Moreover, since Algorithm OIS always outputs the "rightmost" colour-preserving monotone isomorphism from H_i to G, as we explained at the end of Section 3, the vertices $x_{a_1}, \ldots, x_{a_k}$ all belong to G'. In other words, the graph obtained from H by deleting the connected component H_i is isomorphic to an induced subgraph of the graph obtained from G by deleting the vertices $x_{a_1}, x_{a_1+1}, \ldots, x_n$.

The above structural observations suggest the following algorithm for deciding whether or not H is isomorphic to an induced subgraph of G. We start by computing a permutation ordering $\sigma = \langle x_1, \ldots, x_n \rangle$ and a proper 2-colouring f_G for G. We then iterate over all $k!$ permutations $\psi : \{1, \ldots, k\} \to \{1, \ldots, k\}$, corresponding to the $k!$ different arrangements of the connected components of H, and do as follows for each permutation ψ. We consider the connected component H_i of H with $i = \psi(k)$. We run Algorithm OIS on input $(G, f_G, \sigma;\ H_i, f, \omega)$ for all $\omega \in \text{perm}(\tau_i)$ and $f \in \{f_i, f_i'\}$, where τ_i is an arbitrary permutation ordering for H_i and f_i and f_i' are two different 2-colourings for H_i. Recall that there are at most eight such inputs, since $\text{perm}(\tau_i)$ contains at most four vertex orderings. If Algorithm OIS accepts on one of these inputs and outputs index tuple $(a_1, \ldots, a_k)$, we delete the vertices $x_{a_1}, x_{a_1+1}, \ldots, x_n$

from G. We then iterate this process over the remaining connected components H_i of H, decreasing the value of i from $k-1$ down to 1. We accept if and only if Algorithm OIS accepts on input $(G, f_G, \sigma;\ H_1, f, \omega)$ for some $\omega \in \text{perm}(\tau_1)$ and $f \in \{f_1, f_1'\}$.

The correctness of the algorithm immediately follows from the structural observations made at the beginning of the proof. Let us analyse the running time. The algorithm considers $k!$ different arrangements of the connected components of H. For each of these, we run OIS at most eight times for each of the k connected components of H. Recall that OIS runs in polynomial time by Corollary 3.4. This gives an overall running time of $k! \cdot 8k \cdot n^{\mathcal{O}(1)}$, which is equivalent to $k! \cdot n^{\mathcal{O}(1)}$ because of $k \leq n$ and $n \geq 2$.

The proof for the case where G and H are proper interval graphs is similar. In this case, the algorithm again considers all $k!$ permutations $\psi : \{1, \ldots, k\} \to \{1, \ldots, k\}$, but for each of these, it only has to run OIS twice instead of at most eight times for each connected component of H. This yields an overall running time of $k! \cdot 2k \cdot n^{\mathcal{O}(1)}$ in this case. ∎

Our tractability results rely on an efficient algorithm for solving a restricted version of the Induced Subgraph Isomorphism problem. A natural question to ask is whether our approach is applicable to other classes of input graphs. Are there other classes of cocomparability graphs that have "almost-unique" vertex orderings to yield results similar to Lemmas 4.3 and 5.5? As a simple example, it is not difficult to see that complete r-partite graphs in fact admit such a result, when r is bounded. We showed that vertex orderings for our considered graphs are "isomorphic" when they coincide in the first vertex (first statements of Lemmas 4.1 and 5.3). Can we obtain such results for classes of cocomparability orderings when two vertex orderings coincide on more than one position? Does bounding the number of such positions yield fixed-parameter tractable cases of the problem?

We solved our variant of the Induced Subgraph Isomorphism problem by using proper colourings of the input graphs. Our approach worked due to the fact that proper 1-colourings are unique by definition, and proper 2-colourings of connected bipartite graphs are unique up to swapping the two colours. However, this uniqueness property does not easily extend to k-colourings for larger values of k. In practical applications, a colour-preserving isomorphism for coloured graphs may even be the actual desired problem. Such applications are graph-rewriting systems, where graph patterns are replaced by other patterns. Often, the input graphs have labelled, *coloured*, vertices. It is therefore an interesting problem to study which additional restrictions on the Induced Subgraph Isomorphism problem make the problem tractable on input graphs like interval graphs, cographs, permutation graphs.

We end with the following open questions. What is the computational complexity of ISI if G is an interval graph and H is a connected proper interval graph[1], or if G is a permutation graph and H is a connected bipartite permutation graph?

Acknowledgement

We are grateful to an anonymous referee whose helpful comments and suggestions improved the presentation of the results of the paper.

[1]This case was previously claimed to be solvable in polynomial time [11]. Unfortunately, an error in the proof was discovered.

References

[1] R. Belmonte, P. Heggernes, P. van 't Hof. Edge contractions in subclasses of chordal graphs. *Discrete Applied Mathematics*, 160:999-1010, 2012.

[2] A. Brandstädt, V.B. Le, and J. Spinrad, *Graph Classes: A Survey*, SIAM, Philadelphia, 1999.

[3] L. Cai, S. M. Chan, S. O. Chan. Random separation: a new method for solving fixed-cardinality optimization problems. *Proceedings of IWPEC 2006*, Springer LNCS, 4169:239–250, 2006.

[4] D. G. Corneil, H. Kim, S. Natarajan, S. Olariu, A. P. Sprague. Simple linear time recognition of unit interval graphs. *Information Processing Letters*, 55:99–104, 1995.

[5] P. Damaschke. Induced subgraph isomorphism for cographs is NP-complete. *Proceedings of WG 1991*, Springer LNCS, 484:72–78, 1991.

[6] X. Deng, P. Hell, J. Huang, Linear-time representation algorithms for proper circular-arc graphs and proper interval graphs. *SIAM Journal on Computing*, 25:390–403, 1996.

[7] R. G. Downey and M. R. Fellows. *Parameterized Complexity*. Springer, 1999.

[8] D. Eppstein. Subgraph isomorphism in planar graphs and related problems. *Journal of Graph Algorithms and Applications*, 3(3):1–27, 1999.

[9] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-completeness*. W. H. Freeman & Co., 1979.

[10] M. C. Golumbic. *Algorithmic Graph Theory and Perfect Graphs*. Annals of Discrete Mathematics Vol. 57, Elsevier, 2004.

[11] P. Heggernes, D. Meister, Y. Villanger. Induced Subgraph Isomorphism on interval and proper interval graphs. *Proceedings of ISAAC 2010*, Springer LNCS, 6507:399–409, 2010.

[12] L. Ibarra, The clique-separator graph for chordal graphs. *Discrete Applied Mathematics*, 157:1737–1749, 2009.

[13] D. Kratsch and L. Stewart. Domination on cocomparability graphs. *SIAM Journal on Discrete Mathematics*, 6:400–417, 1993.

[14] S. Kijima, Y. Otachi, T. Saitoh, T. Uno. Subgraph isomorphism in graph classes. *Discrete Mathematics*, 312:3164–3173, 2012.

[15] P. J. Looges and S. Olariu. Optimal greedy algorithms for indifference graphs. *Computers & Mathematics with Applications*, 25:15–25, 1993.

[16] D. Marx and I. Schlotter. Cleaning Interval Graphs. *Algorithmica*, 65:275–316, 2013.

[17] D. W. Matula. Subtree isomorphism in $o(n^{5/2})$. *Annals of Discrete Mathematics*, 2:91–106, 1978.

[18] R. M. McConnell and J. P. Spinrad. Modular decomposition and transitive orientation. *Discrete Mathematics*, 201:189–241, 1999.

[19] M. M. Sysło. The subgraph isomorphism problem for outerplanar graphs. *Theoretical Computer Science*, 17:91–97, 1982.

[20] J. Spinrad, A. Brandstädt, L. Stewart. Bipartite permutation graphs. *Discrete Applied Mathematics*, 18:279–292, 1987.