UNIVERSITE D'ORLEANS

*Faculté des Sciences*

**LIFO**

Laboratoire d'Informatique Fondamentale d'Orléans
4, rue Léonard de Vinci, BP 6759
F-45067 Orléans Cedex 2
FRANCE

Rapport de Recherche

www : http://www.univ-orleans.fr/SCIENCES/LIFO/

# Minimal interval completions

Pinar HEGGERNESS
University of Bergen, Norway
Karol SUCHAN
LIFO, Université d'Orléans
Ioan TODINCA
LIFO, Université d'Orléans
Yngve VILLANGER
University of Bergen, Norway

Rapport N$^o$ **2005-04**

# Minimal interval completions

Pinar Heggernes[1], Karol Suchan[2], Ioan Todinca[2], and Yngve Villanger[1]

[1] Department of Informatics, University of Bergen, N-5020 Bergen, Norway
pinar@ii.uib.no, yngvev@ii.uib.no
[2] LIFO, Université d'Orleans, PB 6759, F-45067 Orleans Cedex 2, France
todinca@lifo.univ-orleans.fr, suchan@lifo.univ-orleans.fr

**Abstract.** We study the problem of adding edges to an arbitrary graph so that the resulting graph is an interval graph. Our objective is to add an inclusion minimal set of edges, which means that no proper subset of the added edges can result in an interval graph when added to the original graph. This problem is closely related to the problem of adding an inclusion minimal set of edges to a graph to obtain a chordal graph, which is a well studied problem and which motivates an analogous study of minimal interval completions. However, whereas there are nice properties that result in efficient algorithms for obtaining a chordal graph in this way, the same problem for obtaining an interval graph is more difficult, and its computational complexity has been open until now. In this paper we give a polynomial time algorithm to obtain a minimal interval completion of an arbitrary graph, thereby resolving the complexity of this problem.

## 1 Introduction

The class of interval graphs is an important and well-studied graph class with applications in many fields, like biology, chemistry, and archeology [5]. In addition, many problems that are NP-complete on general graphs have polynomial-time, and even linear-time, algorithms on interval graphs. Thus it is of interest to compute an interval embedding of a given graph with few added edges. An interval graph can be obtained from any graph by adding edges, and the resulting graph is called an *interval completion* of the original graph. If the added set of edges is of minimum cardinality, the resulting interval completion is called *minimum*. Unfortunately, computing minimum interval completions is an NP-hard problem [3], [6]. The *pathwidth* problem is concerned with computing an interval completion in which the size of the largest clique is minimized. Many difficult problems have efficient solutions for graphs of bounded pathwidth; however also this problem is NP-hard [7]. This has given motivation to study the problem of adding an inclusion minimal set of fill edges to a given graph to obtain an interval completion. That is, no proper subset of the added edges can give an interval completion of the original graph. In this case, the resulting interval completion is called *minimal*. Interval completions with minimum number of edges or with minimum clique size are among the minimal interval completions of the input graph. The computational complexity of minimal interval completions has been open until now. In this paper, we show that the problem can be solved in polynomial time.

Interval graphs are a subset of chordal graphs, and they are exactly the class of graphs that are both chordal and AT-free [10]. Adding edges to an arbitrary graph so that the resulting graph is chordal, gives a chordal completion, or a *triangulation* of the input graph. Minimum and minimal triangulations are defined analogous to minimal and minimum interval completions. Computing minimum triangulations is NP-hard [13], whereas computing minimal triangulations is a well studied problem, and it motivates the study of minimal interval completions [8]. A triangulation is minimal if and only if no single added edge can be removed without destroying chordality [12]. This useful property gives algorithmic tools that have been used in several minimal triangulation algorithms. Unfortunately, the analogous statement for interval completions is not true. That is, it might be the case that no single added edge can be removed from an interval completion without destroying the interval graph property, but still the interval completion might not be minimal because there are several edges that can be removed simultaneously to give a minimal interval completion. Furthermore, examples can be constructed to show that a minimal triangulation followed by a minimal AT-free completion, or vice versa, does not necessarily result in a minimal interval completion. Consequently, the minimal interval completion problem is more difficult than the minimal triangulation problem, and it was not known until now whether minimal interval completions could be computed in polynomial time.

We use an incremental approach for some fixed ordering of vertices to compute a minimal interval completion of a given arbitrary graph $G$: At each step, a new vertex is taken into account and we compute a minimal interval completion of the subgraph of $G$ induced by the vertices considered so far. For each new vertex $u$ processed in this way, edges are added only between $u$ and the vertices processed before. The vertices of $G$ can be processed in any order, and a step does not require modifications of the previously added edges or knowledge about the parts of $G$ that are not yet considered. Thus, our algorithm can be implemented as an on-line algorithm where vertices of $G$ are supplied one by one together with their neighborhoods in the already processed part of $G$. A similar approach was used for minimal triangulations in [1]. In addition, we use the property of interval graphs that there is a natural linear order of their maximal cliques. The main obstacle is to find a linear order of the original graph that corresponds to such a linear order of the maximal cliques of the resulting minimal triangulation. We overcome this problem by using several techniques, like partition refinement and Dilworth decomposition of an order into paths. The overall time complexity of our algorithm is $O(n^5)$.

## 2 Preliminaries

We start this section with standard definitions, and at the end of this section we give new definitions specific to our problem and our algorithm.

We consider simple and connected input graphs. A graph is denoted by $G = (V, E)$, with $n = |V|$, and $m = |E|$. For a set $A \subseteq V$, $G[A]$ denotes the subgraph of $G$ induced by the vertices in $A$. Vertex set $A$ is called a *clique* if $G[A]$ is complete. For a vertex $v \in V$ or a subset $A \subseteq V$, we will informally use $G - v$ and $G - A$ to denote the graphs $G[V \setminus \{v\}]$ and $G[V \setminus A]$, respectively. A path is a sequence $[v_1, v_2, \ldots, v_p]$ of vertices such that $v_i$ is adjacent to $v_{i+1}$, for all $1 \leq i < p$. A cycle is a path such that the first and last vertices are adjacent.

The *neighborhood* of a vertex $v$ in $G$ is $N_G(v) = \{u \mid uv \in E\}$, and the *closed neighborhood* of $v$ is $N_G[v] = N_G(v) \cup \{v\}$. Similarly, for a set $A \subseteq V$, $N_G(A) = \bigcup_{v \in A} N_G(v) \setminus A$, and $N_G[A] = N_G(A) \cup A$. When graph $G$ is clear from the context, we will omit subscript $G$.

An edge that is added to the input graph $G$ is called a *fill edge*, and the process of adding edges between a vertex $x$ and a vertex set $A$ is called *filling* $A$.

A vertex set $S \subset V$ is a *separator* if $G - S$ is disconnected. Given two vertices $u$ and $v$, $S$ is a $u, v$*-separator* if $u$ and $v$ belong to different connected components of $G - S$, and $S$ is then said to *separate* $u$ and $v$. A $u, v$-separator $S$ is *minimal* if no proper subset of $S$ separates $u$ and $v$. In general, $S$ is a *minimal separator* of $G$ if there exist two vertices $u$ and $v$ in $G$ such that $S$ is a minimal $u, v$-separator. It can easily be verified that $S$ is a minimal separator if and only if $G - S$ has two distinct connected components $C_1$ and $C_2$ such that $N_G(C_1) = N_G(C_2) = S$. In this case, $C_1$ and $C_2$ are called *full components*, and $S$ is a minimal $u, v$-separator for *every* pair of vertices $u \in C_1$ and $v \in C_2$.

A *chord* of a cycle is an edge connecting two non-consecutive vertices of the cycle. A graph is *chordal*, or equivalently *triangulated*, if it contains no induced chordless cycle of length $\geq 4$.

A graph $G$ is an *interval* graph if continuous intervals can be assigned to each vertex of $G$ such that two vertices are neighbors if and only if their intervals intersect. In other terms, interval graphs are the intersection graphs of subpaths of a path.

Every interval graph is also chordal. A graph is an interval graph if and only if it is chordal and AT-free [10].

**Theorem 1 ([4]).** *A graph $G$ is interval if and only if there is a path $(\mathcal{K}, P)$ whose vertex set is the set of all maximal cliques of $G$, such that the subgraph of $(\mathcal{K}, P)$ induced by the maximal cliques of $G$ containing vertex $v$ is connected, for each vertex $v$ of $G$.*

Such a path will be called a *clique path* $CP_G$ of $G$. The vertices of a clique path (maximal cliques of $G$) are also called *bags*. Let the maximal cliques of an interval graph $G$ be labelled $1, 2, ..., k$, according to the order in which they appear in a clique path of $G$. Then, as a consequence of Theorem 1, an interval representation of $G$ can be obtained by assigning to each vertex $v$ the interval that consists of the labels of the maximal cliques containing $v$. In this way, every clique path of $G$ is equivalent to an interval representation of $G$.

**Lemma 1 (see e.g. [5]).** *Let $G$ be an interval graph and let $CP_G$ be any clique path of $G$. A set of vertices $S$ is a minimal separator of $G$ if and only if $S$ is the intersection of two maximal cliques of $G$ that are neighbors in $CP_G$. In particular, all minimal separators of $G$ are cliques.*

We now give some definitions particularly related to our results. Let $S$ and $T$ be two minimal separators of $G$ such that no one of them is a subset of the other. Suppose that $T$ (resp. $S$) intersects a unique component $C_T(S)$ (resp. $C_S(T)$) of $G - S$ (resp. $G - T$) and that $C_T(S)$ (resp. $C_S(T)$) is full for $S$ (resp. $T$). We define the *piece between* $S$ and $T$ by $P(S,T) = S \cup T \cup (C_T(S) \cap C_S(T))$.

A *block* $B$ of $G$ is a vertex subset such that $B$ is the piece between $P(S,T)$ for some minimal separators $S, T$ (in which case we say that $B$ is a *two-block*), or $B = S \cup C$ for some minimal separator $S$ and a full component $C$ of $G - S$ (in this case $B$ is a *one-block*). We say that the separators $S$ and $T$ (respectively $S$) *border* the block $B$.

**Lemma 2.** *Let $B$ be a block of an interval graph $G$. Then $B$ is the union of maximal cliques of $G$ contained in $B$.*

*Proof.* For any $x \in B$ that is not on a bordering separator, any maximal clique containing $x$ is contained in $B$. If $S$ is a minimal separator bordering $B$, there is a maximal clique $K$ of $G$ with $S \subset K \subseteq B$ (see e.g. Lemma 4.6 of [2]). □

Let us note that, for any block $B$ of an interval graph $G$, and for any clique path $CP_G$ of $G$, the maximal cliques contained in $B$ form a (connected) subpath of $CP_G$.

**Lemma 3.** *Let $B$ be a block of an interval graph $G$ and let $CP_G = (\mathcal{K}, P)$ be any clique path of $G$. The set of bags corresponding to maximal cliques contained in $B$ form a connected subpath $P_B$ of $P$.*

*Proof.* First we prove the lemma when $B$ is a one-block $S \cup C$. Assume on the contrary that two maximal cliques $K_1$ and $K_2$ contained in $B$ are separated in $CP_G$ by a maximal clique $K \not\subseteq B$. By properties of a clique path, $K$ separates in $G$ any two vertices $a \in K_1 \setminus K$ and $b \in K_2 \setminus K$. Then $K$ must intersect $C$ in a vertex $c$, so $K \subseteq N[c]$ *subseteqB*, which is a contradiction.

If $B$ is a two-block $P(S,T)$, let $C_T(S)$ be the connected component of $G - S$ intersecting $T$. The one-block $B' = S \cup C_T(S)$ corresponds do a subpath $P_{B'}$ of $CP_G$, and this subpath is a clique path of $G[B']$. Note that $T$ is a minimal separator of $G[B']$ and $B$ is a one-block of $G[B']$. Thus the cliques of $B$ form a subpath of $P_{B'}$ and the conclusion follows. □

Clearly for any clique path $CP_G$ of $G$, its subpath $P_B$ defines a clique path of $G[B]$. The converse is not true, there are clique paths of $G[B]$ which can not be extended into clique paths of $G$. Consider the case when $B$ is a one-block. The next lemma characterizes the clique paths of $B$ extendable into clique paths of the whole graph.

**Lemma 4.** *Let $B$ be a block of an interval graph $G$.*

*1. A clique path $CP_{G[B]}$ of $G[B]$ can be extended into a clique path of $G$ if and only if each separator bordering $B$ is contained in a maximal clique that is a (different) endpoint of $CP_{G[B]}$.*

*2. In a clique path $CP_G$ of $G$, subpath $P_B$ can be replaced by any clique path of $G[B]$ satisfying the above property.*

*Proof.* Let $CP_{G[B]}$ correspond to a subpath of $CP_G$. We consider first the case when $B$ is a one-block $S \cup C$. Let $D$ be another full component of $G - S$ associated to $S$ and $B_D = S \cup D$. Then $P_B$ and $P_{B_D}$ are disjoint subpaths of $CP_G$. If $P_{B_D}$ is to the right (resp. left) of $P_B$, by the properties of a clique path the leftmost (resp. rightmost) clique of $P_B$ must contain $B \cap B_D = S$ and the conclusion follows. The case $B = P(S, T)$ is similar, observing like in Lemma 3 that $B$ is a one-block in the graph $G[B']$, where $B'$ is the one-block $S \cup C_T(S)$ associated to $S$ containing $T$.

Conversely, let $CP_{G[B]}$ be any clique path of $G[B]$ such that the separators bordering $B$ are contained in the end cliques. For any clique path $CP_G$ of $G$, we can replace the subpath $P_B$ formed by the cliques of $B$ with $CP_{G[B]}$ without violating the properties of a clique path. □

Let us observe that a minimal interval completion can be obtained incrementally.

**Lemma 5.** *Let $H$ be a minimal interval completion of an arbitrary graph $G$. Let $G'$ be a graph obtained from $G$ by adding a new vertex $x$, with neighborhood $N_{G'}(x)$. There is a minimal interval completion $H'$ of $G'$ such that $H' - x = H$.*

*Proof.* Let $H''$ be the graph obtained by adding $x$ and the edges between $x$ and $N_{G'}(x)$ to $H$. Observe first that an interval completion of $H''$ can be obtained by adding edges only incident to $x$. For example, we can make $x$ adjacent to all vertices of $H''$. Clearly the result is an interval graph, since this corresponds to adding a new interval to the interval representation of $H$ such that the new interval, corresponding to vertex $x$, has a smaller starting point and a larger ending point than any of the other intervals. If the resulting graph is not a minimal interval completion of $H''$, a suitable set of the newly added edges between $x$ and vertices of $H''$ outside of $N_{G'}(x)$ can be removed to obtain a minimal interval completion.

Let $H'$ be a minimal interval completion of $H''$ obtained by adding edges incident to $x$. We claim that $H'$ is a minimal interval completion of $G'$. To see this, observe that the interval graph property is hereditary, so that any induced subgraph of an interval graph is also interval. Now, if we assume that a subset of the newly added edges to $H''$ to obtain $H'$ and a subset of the edges previously added to $G$ to obtain $H$ can be removed from $H'$ without destroying the interval property, we reach a contradiction since then $H$ cannot be a minimal interval completion of $G$. □

Hence, for computing a minimal interval completion of $G$, we introduce the vertices of $G$ one by one in the order $x_1, x_2, \ldots, x_n$. Given a minimal interval completion of $H_i$ of $G_i = G[\{x_1, \ldots, x_i\}]$, we compute an interval completion $H_{i+1}$ of $G_{i+1}$ by adding vertex $x_{i+1}$ and the edges between $x_{i+1}$ and $N_{G_{i+1}}$ to $H_i$, together with a well chosen set of additional edges incident to $x_{i+1}$.

## 3 Principles of the algorithm

From now on we consider as input an interval graph $G = (V, E)$. A new vertex $x$ is added to $G$, together with a set of edges incident to $x$. For the rest of this document, let $G'$ denote the graph $G + x$. We want to compute a minimal interval completion $H$ of $G'$, obtained by adding edges incident to $x$ only. We say that such a minimal interval completion *respects* $G$.

Take any clique path $CP_H = (\mathcal{K}, P)$ of $H$. By property of clique paths, the cliques containing $x$ form a subpath $P_x$ of $P$. Now, let us get back to $G$. Delete $x$ from every bag in $CP_H$, and possibly remove the bags that do not correspond to maximal cliques of $G$. This yields $CP_G$ - a clique path of $G$, which gives a linear ordering of maximal cliques of $G$. We say in this case that $CP_G$ was obtained by *pruning* the vertex $x$ from $CP_H$.

**Definition 1.** *An interval completion $H$ of $G'$* respects *a clique path $CP_G$ of $G$ if $CP_G$ can be obtained by pruning $x$ from some clique path of $H$.*

Clearly the maximal cliques that come from $P_x$ still form a subpath of $CP_G$. Our aim is to do the converse: to find a clique path $CP_G = (\mathcal{K}, P)$ of $G$ and a subpath of $CP_G$ in which, by adding vertex $x$ to every bag and possibly transforming the bordering separators into new bags of $H$ (with $x$ contained), we obtain a minimal interval completion of $G'$ respecting $CP_G$.

A clique path $CP_G$ is called *n*ice if there exists a minimal interval completion $H$ respecting $CP_G$. Let $K_L$ (resp. $K_R$) be the leftmost (resp. rightmost) clique of $CP_G$ such that $x$ has a neighbor in $K_L \setminus K_{L+1}$ (resp. $K_R \setminus K_{R-1}$) in the graph $G'$. Any minimal interval completion $H$ respecting $CP_G$ satisfies:

- $xu \in E(H)$ for every vertex $u$ contained in a clique strictly between $K_L$ and $K_R$ of $CP_G$.
- $xv \notin E(H)$ for every vertex $v$ that is not contained in any clique situated between $K_L$ and $K_R$ in the clique path.

We point out that, it some cases, it might be required to add fill edges between $x$ and some vertices of $K_L$ or $K_R$. Note that any clique path obtained from a nice one by permutations of the cliques strictly between $K_L$ and $K_R$, or outside the interval between $K_L$ and $K_R$ will also be a nice clique path of $G$. We will not look for one particular clique path, but a class of them, yielding the same interval completion. Working towards finding a nice clique path of $G$ and cliques $K_L$ and $K_R$, we refine ordered partitions of the set $\mathcal{K}$ of all maximal cliques of $G$.

**Definition 2.** *Consider an ordered partition $OP = [\mathcal{K}_1, \ldots, \mathcal{K}_p]$ of $\mathcal{K}$ such that the maximal cliques in each part $\mathcal{K}_i$ correspond to a block of $G$ and, moreover, there exists a clique path $CP_G$ of $G$ such that the subpaths $P_1, \ldots, P_p$ formed by the maximal cliques in $\mathcal{K}_1, \ldots \mathcal{K}_p$ appear in this order (see also Lemma 3). Such an ordered partition is called* valid.

Observe that, when each part of a valid ordered partition $OP$ is a singleton, $OP$ is exactly a clique path of $G$. Here, again we say that the ordered partition $OP$ is *nice* if it can be refined into a nice clique path $CP_G$.

*Remark 1.* Consider a nice ordered partition $OP$. Let $\mathcal{K}_L$ (resp $\mathcal{K}_R$) be the leftmost (resp. rightmost) part of $OP$ such that $x$ has a neighbor appearing in $\mathcal{K}_L$ but not in $\mathcal{K}_{L+1}$ (resp. appearing in $\mathcal{K}_R$ but not in $\mathcal{K}_{R-1}$). For any minimal interval completion $H$ of $G'$ respecting some clique path $CP_G$ obtained by refining $OP$ we have:

- $xu \in E(H)$ for every vertex $u$ appearing in a part strictly between $\mathcal{K}_L$ and $\mathcal{K}_R$ of $OP$
- $xv \notin E(H)$ for every vertex $v$ not appearing in $\mathcal{K}_L, \mathcal{K}_R$ or in any other part situated between $\mathcal{K}_L$ and $\mathcal{K}_R$ in the ordered partition.

Once we obtain a nice ordered partition $OP$ and parts $\mathcal{K}_L$ and $\mathcal{K}_R$, it remains to add a well chosen set of edges between $x$ and blocks $O_L$ and $O_R$, corresponding to the sets of maximal cliques $\mathcal{K}_L$ and $\mathcal{K}_R$ respectively. This will be done by refining recursively the sets of cliques $\mathcal{K}_L$ and $\mathcal{K}_R$.

For obtaining the nice ordered partition $OP$ we distinguish between two cases:

*Case 1: The neighborhood of $x$ in $G$ is a clique.* In this situation we show that, if $G'$ is not already an interval graph, is it sufficient to add edges from $x$ to a well chosen minimal separator of $G$, see Theorem 11 in Appendix B.

*Case 2: The neighborhood of $x$ in $G$ is not a clique.* This is the main case of our algorithm. Let $\mathcal{S}_x$ be the set of all minimal $a,b$-separators of $G$, for all non-adjacent $a, b \in N_{G'}(x)$. Let $S_L, S_R$ be a pair of separators in $\mathcal{S}_x$ forming a block $B = P(S_L, S_R)$ (or $B = S \in \mathcal{S}_x$ if $S$ is inclusion maximum in $\mathcal{S}_x$). We shall see that for any interval completion $H$ respecting $G$, the block $B = P(S_L, S_R)$ is filled in $H$, i.e. $B \subseteq N_H(x)$. We define the one-blocks associated to $B$ as the blocks of the form $C_i \cup N_G(C_i)$, for each connected component $C_i$ of $G - B$. These one-blocks and $B$ induce a partition of the cliques of $G$, that we use for obtaining a nice ordered partition.

## 4 The main case

Let us consider the main and most difficult case of our algorithm, when the neighborhood of $x$ in $G'$ is not a clique. We denote by $\mathcal{S}_x$ the set of all minimal $a,b$-separators of $G$, with $a, b \in N_{G'}(x)$ and non-adjacent. Clearly $\mathcal{S}_x$ is not empty.

The following theorem states that, for any $S \in \mathcal{S}_x$, we must add all the edges from $x$ to $S$.

**Theorem 2 ([1]).** *Let $H$ be any chordal supergraph of $G'$ such that $H - x = G$. Consider a minimal $a,b$-separator $S$ of $G$, where $a$ and $b$ are neighbors of $x$ in $G'$. Then $S$ is in the neighborhood of $x$ in $H$.*

This observation can be extended to any piece between two elements of $\mathcal{S}_x$.

**Theorem 3.** *Let $H$ be any interval supergraph of $G'$ such that $H - x = G$. Any block $B = P(S_L, S_R)$ that is a piece between two separators in $\mathcal{S}_x$, is contained in the neighborhood of $x$ in $H'$.*

*Proof.* The bordering separators $S_L$, $S_R$ are incomparable by inclusion. On the other hand, they both have at least two full components containing vertices adjacent to $x$, only one of which can have a nonempty intersection with $B$. Let $C_L$ ($C_R$) denote a full component of $G - S_L$ ($G - S_R$ respectively) containing a neighbor of $x$ and having empty intersection with $B$. Let $O_L = C_L \cup S_L$ and $O_R = C_R \cup S_R$. By Lemma 3, in any clique path, blocks $B$, $O_L$ and $O_R$ form connected subpaths. Since $S_L$, $S_R$ are incomparable by inclusion, subpaths $P_{O_L}$ and $P_{O_R}$ have to be on different sides of $P_B$. Indeed if $P_{O_R}$ is between $P_{O_L}$ and $P_B$, then $S_L = O_L \cap B$ is contained in every maximal clique of $O_R$. In particular $S_L$ (and thus $B$) intersects $C_R$, which is a contradiction. So there have to be cliques containing $x$ both to the left and right of $P_B$. Thus, by property of clique paths, every clique of $B$ should have vertex $x$ added. □

From now on we fix a block $B = P(S_L, S_R)$ with $S_L, S_R \in \mathcal{S}_x$, if such a block exists. For each component $C_j$ of $G - B$, let $O_j = N(C_j) \cup C_j$. Note that $N(C_j)$ is a minimal separator, so $O_j, j = 1, \ldots, k$ are full one-blocks associated to separators that are subsets of $S_L$ or $S_R$. Let $\mathcal{O}(B)$ denote the set of all these one-blocks associated to $B$, and $\mathcal{B}$ the set $\mathcal{O} \cup \{B\}$. Notice that the blocks $\mathcal{B}$ induce a partition of the set $\mathcal{K}$ of all maximal cliques of $G$. Indeed, each $K \in \mathcal{K}$ is contained in exactly one element of $\mathcal{B}$. Moreover, by Lemma 3, for any clique path $CP_G = (\mathcal{K}, P)$ of $G$, the cliques contained in a block $BL \in \mathcal{B}$ induce a subpath of $CP_G$. Therefore, there is a natural correspondence between clique paths and valid ordered partitions of set $\mathcal{K}$ induced by $\mathcal{B}$.

Consider such a valid ordered partition $OP = [\mathcal{K}_1, \dots, \mathcal{K}_p]$ corresponding to clique path $CP_G$. Instead of working with the ordered partition $OP$, we work with the pair $(L, R)$ where $L$ is the list of one-blocks of $\mathcal{O}(B)$ situated to the left of $B$ in the clique path, ordered from left to right; $R$ is the list of blocks to the right of $B$, from right to left. More formally, let $\mathcal{K}_i$ be the part corresponding to block $B$ (i.e., the elements of $\mathcal{K}_i$ are exactly the maximal cliques of $B$). Then $L = [O_1, O_2, \dots, O_{i-1}]$ and $R = [O_p, O_{p-1}, \dots, O_{i+1}]$, where $O_j$ is the one-block corresponding to part $K_j$. Clearly the pair $(L, R)$ identifies the ordered partition $OP$. Now given two lists $L$ and $R$ such that each one-block of $\mathcal{O}(B)$ is in exactly one list, we want to know if the pair $(L, R)$ defines a valid ordered partition.

*Remark 2.* Graph $G$ might not contain a block $B = P(S_L, S_R)$ as defined above. In this case, let $B$ be any minimal separator of $G$ contained in $\mathcal{S}_x$. According to Lemma 1, $B$ corresponds to some edge of the clique path $CP_G$. We have to choose the edge such that $x$ has neighbors both to the left and to the right of it, which is always possible. (We could consider that the subpath $P_B$ is "degenerated"). Here again we can define $\mathcal{O}(B)$ and the ordered partitions $(L, R)$ as before, with the small difference that $\mathcal{O}(B)$ covers now all the maximal cliques of $G$. All the statements proved in the rest of the paper remain true when $B = P(S_L, S_R)$ is replaced by $B = S$. For sake of readability, we do not discuss this situation as a special case. Note that, for our algorithm, it is more convenient (faster) to use a block rather than a separator if possible.

Consider two one-blocks $O_i$ and $O_j$ in $\mathcal{O}(B)$. We want to know whether there exists a clique path $CP_G$ of $G$ in which the subpath $P_{O_j}$ is between $P_{O_i}$ and $P_B$. The answer is yes only if $O_i \preceq O_j$ according to the pre-order defined as follows:

**Definition 3.** *For every $O \in \mathcal{O}(B)$ let*

$$big(O) = \max\{K \cap B | K \in \mathcal{K}, K \subseteq O\}, \quad small(O) = \min\{K \cap B | K \in \mathcal{K}, K \subseteq O\} \tag{1}$$

*Two one-blocks $O_i, O_j$ satisfy $O_i \preceq O_j$ if $big(O_i) \subseteq small(O_j)$.*

**Lemma 6.** *Let $O_i$ and $O_j$ be two one-blocks of $\mathcal{O}(B)$. There is a clique path $CP_G$ of $G$ such that the subpath $P_{O_j}$ is between $P_{O_i}$ and $P_B$ only if $O_i \preceq O_j$.*

It is actually more convenient to prove the following stronger statement. It shows that this pre-order captures all possible clique paths of $G$.

**Theorem 4.** *Let $L, R$ be two lists such that each block of $\mathcal{O}(B)$ appears in exactly one of them. Then $(L, R)$ defines a valid ordered partition if and only if $L$ and $R$ are directed paths of partially pre-ordered set $(\mathcal{O}(B), \preceq)$.*

*Proof.* For the "only if" part, note that for two blocks $O_i$ and $O_j$ there exist a clique path in which $P_{O_j}$ is between $P_{O_i}$ and $P_B$ only if $O_i \cap B$ is contained in every maximal clique of $G[O_i]$. Hence $L$ and $R$ must be chains of $(\mathcal{O}(B), \preceq)$.

Conversely, if we partition $(\mathcal{O}(B), \preceq)$ into two chains $(L, R)$ we can construct a clique path of $G$ in which the subpaths corresponding to one-blocks in $L$ (resp. $R$) are situated to the left (resp. right) of $P_B$, respecting the order of the chains $L$, $R$. In this setting, we only need to get a clique path for each block $BL \in \mathcal{B}$ that lets us glue them all together into a clique path of $G$.

By using Lemma 4, we can construct a clique path of every $O \in \mathcal{O}(B)$ with $big(O)$ at one end. In a similar way we can get a clique path of $B$ with separators $S_L$ and $S_R$ bordering $B$ in $G$ at the ends. The conclusion follows. $\square$

Our aim is to construct ordered partitions that are not only valid, but also nice. Unfortunately the pre-order $(\mathcal{O}(B), \preceq)$ is not enough to manage nice ordered partitions (and thus minimal interval completions). The reason is that, for one-blocks that are equivalent with respect to $\preceq$, their placement is not without impact on the minimality of the interval completion obtained.

**Definition 4.** *We say that a one-block $O$ is:*

- clean *if $O \cap N_{G'}(x) \subset B$*
- hit *if it is not clean*
- sparable *if there is an interval completion of $G'_d[O \cup \{x\}]$, constructed form $G'[O \cup \{x\}]$ by adding a dummy vertex $d$ adjacent to $x$ and to all the vertices in the separator bordering $O$, in which $x$ is non adjacent to at least one vertex of $O$.*

In particular, all clean blocks are sparable.

**Definition 5.** *Let $(L, R)$ be any valid order partition of $G$. An interval completion $H(L, R)$ is* good *for $(L, R)$ if it respects some clique path obtained by refining $(L, R)$ and it is inclusion minimal for this property.*

The following proposition characterizes the best interval completions that can be obtained from a valid ordered partition $(L, R)$.

**Lemma 7.** *Let $(L, R)$ define a valid ordered partition. Let $O_L$ and $O_R$ be the first hit blocks of $L$ and $R$ respectively*[1]*. An interval completion $H$ of $G'$ is a good for $(L, R)$ if and only if:*

1. *For each block $O$ appearing after $O_L$ in $L$ or after $O_R$ in $R$, $O$ is filled.*
2. *For each block $O$ appearing before $O_L$ in $O$ or before $O_R$ in $R$, $O$ stays clean in $H$.*
3. *For each $O \in \{O_L, O_R\}$, the graph $H_d[O \cup \{x\}]$ is a minimal interval completion of $G'_d[O \cup \{x\}]$. Here $H_d[O \cup \{x\}]$ is obtained from $H[O \cup \{x\}]$ by adding a dummy vertex $x$ adjacent to $x$ and all the vertices in the separator bordering $O$ (see also Definition 4).*

*Proof.* Let $H$ be a good interval completion for $(L, R)$ and consider a clique path $CP_G$ obtained by refining $(L, R)$.

By the fact that $O_L$ and $O_R$ are hit, there are $y_L \in O_L$ and $y_R \in O_R$, adjacent to $x$ in $G'$ and not appearing in any other block. Therefore $x$ must be added to some bag of $O_L$ and of $O_R$, thus it will be added to all bags of the blocks between $O_L$ and $O_R$ in the clique path. Note that this first point holds for any interval completion respecting $CP_G$.

For the second point, consider any interval completion $H'$ respecting $CP_G$. By removing $x$ from all the bags situated to the left of $O_L$ or to the right of $O_R$, we still obtain an interval completion of $G'$.

Eventually, suppose that $H_d[O \cup \{x\}]$ is not a minimal interval completion of $G'_d[O \cup \{x\}]$, for some $O \in \{O_L, O_R\}$, so it strictly contains an interval completion $H'$. Denote by $S$ the minimal separator bordering $O$ in $G$. Note that $S \cup \{x\}$ is a minimal separator of $H'$ and of $H$. Hence, $O \cup x$ is a one-block in $H$ and $H'$. By Lemma 4, we can replace the subpath corresponding to $O \cup \{x\}$ in a clique path of $H$ by the similar subpath from the clique path of $H'$. The new interval graph is strictly contained in $H$, contains all the edges between $x$ and $O$ and respects a clique path obtained by refining $(L, R)$, contradicting the minimality of $H$.

Conversely, if $H$ is an interval completion respecting the three properties, it is necessarily good for $(L, R)$. As noticed above all interval respecting $(L, R)$ satisfy the first point. Also, for

[1] If $O_R$ is not defined, imagine it as added at the end of the list $R$; the set of blocks appearing after $O_R$ in $R$ becomes empty, the set of blocks appearing before it is exactly $R$. The case when $O_L$ is undefined is symmetrical.

any interval completion $H'$, $H'_d[O \cup \{x\}]$ is an interval completion of $G'_d[O \cup \{x\}]$, for any block $O \in \mathcal{O}(B)$. Indeed it is sufficient to note that $O \cup \{x\}$ is a one-block of $H'$ bordered by $S \cup \{x\}$, so the subpath induced by this block in a path decomposition of $H'$ can be extended by adding the clique $S \cup \{d, x\}$ at one end (by Lemma 4). □

In particular, if $H$ is a good interval completion for $(L, R)$ and $O \in \{O_L, O_R\}$ is sparable, then it is not filled in $H$.

**Theorem 5.** *Let $(L, R)$ define a valid ordered partition. Denote by $Spared(L, R)$ the set of one-blocks appearing before $O_L$ in $L$, before $O_R$ in $R$ or being in $\{O_L, O_R\}$ and sparable. The valid ordered partition $(L, R)$ is nice if and only if $Spared(L, R)$ is inclusion-maximal.*

*Proof.* According to Lemma 7, the set $Spared(L, R)$ is exactly the set of blocks that may not be filled in an interval completion respecting $(L, R)$.

Let $(L, R)$ be such that $Spared(L, R)$ is maximal and $H$ be a good interval completion for $(L, R)$. Suppose that $H$ is strictly contained in some interval completion $H'$ of $G'$. Denote by $(L', R')$ the valid ordered partition corresponding to some path decomposition of $H'$. Let $O \in \mathcal{O}(B)$ such that $N_{H'}(x) \cap O$ is strictly contained in $N_H(x) \cap O$. Note that $O$ is in $Spared(L, R)$, by maximality of this set. If $O$ is a hit block (i.e. $x$ has a neighbor in $O \setminus B$ in the graph $G$) then $O \in \{O_L, O_R\}$, contradicting Lemma 7. If $O$ is not hit, then $N_H(x) \cap O$ is exactly the separator bordering $O$. By Theorem 3, this separator is filled in any interval completion of $G'$, leading again to a contradiction.

Conversely, let $H$ be a minimal interval completion of $G'$. Suppose there exists a valid order partition $(L', R')$ with $Spared(L, R) \subset Spared(L', R')$. Then all the blocks that are clean in $H$ are also clean in any interval completion $H'$ good for $(L', R')$. If $O$ is sparable but hit in $(L, R)$, then $O$ is $O_L$ or $O_R$. Thus $O$ is either $O_{L'}$ or $O_{R'}$. By Lemma 7 we can take $H'$ such that $H'[O] = H[O]$. Consequently $H \subseteq H'$. Choose now a block $O$ in $Spared(L', R') \setminus Spared(L, R)$, so $O$ is not filled in $H'$. We obtain that $H'$ is strictly contained in $H$, a contradiction. □

Suppose that $O_1$, $O_2$, $O_3$ are equivalent with respect to $\preceq$, and that they represent all one-blocks that should be placed to the left of $B$. Let $O_1$ be clean, $O_2$ hit but sparable, and $O_3$ not sparable. If we put them in any order different than $1-2-3$, then the edges we add form a superset of those added by order $1-2-3$, following Theorem 5.

Let $(\mathcal{O}(B), \leq)$ denote the pre-order $(\mathcal{O}(B), \preceq)$ refined to incorporate this differentiation.

**Definition 6.** *For any $O_i, O_j \in \mathcal{O}(B)$,*
*$O_i \leq O_j$ if $O_i \preceq O_j$ and (not $O_j \preceq O_i$ or priority$(O_i) \leq$ priority$(O_j)$), where*

$$priority(O) = \begin{cases} 1, \text{ if } O \text{ is clean;} \\ 2, \text{ if } O \text{ is hit but sparable;} \\ 3, \text{ if } O \text{ is not sparable.} \end{cases}$$

Note that one can check in polynomial time if a one-block is sparable.

**Lemma 8.** *A block $O$ is sparable if and only if there exists a vertex $y \in O \setminus (B \cup N_{G'}(x))$ such that the graph obtained from $G'_d[O \cup \{x\}]$ by filling $O \setminus \{y\}$ is an interval graph. It can be checked in $O(nm)$ time if a given one-block $O \in \mathcal{O}(B)$ is sparable.*

*Proof.* Assume that $O$ is sparable. Let $H$ be an interval completion of $G'_d[O \cup \{x\}]$ in which $O$ is not filled, and let $CP_H$ be its clique path. Note that $O + x$ is a one-block of $G'_d[O \cup \{x\}]$. By Lemma 3, the clique $\{d\} \cup N(d) \cup \{x\}$ is at one of the ends of $CP_H$. The clique $K$ at the other and is not filled. Pick a vertex $y \in K$ that does not belong to any other clique. Making all vertices but $y$ adjacent to $x$ yields an interval completion as needed. The converse is straightforward and the conclusion follows. $\square$

The algorithm **NiceOrderedPartition** of Table 1 produces a nice ordered partition $(L, R)$. The full proof of the algorithm is moved to Appendix A. Let us give here a description of the algorithm and some hints about its proof.

The algorithm relies on the pre-order $\leq$. We transform it into a linear order $\leq_{top}$ by sorting the one-blocks in topological order (the permutation of elements in the same equivalence class does not affect the minimality of interval completions obtained). The graph $(\mathcal{O}, \preceq)$ together with a topological ordering $\leq_{top}$ of the refined pre-order is taken as input. The set $\mathcal{O}(B)$, the pre-orders and the topological order have to be computed before launching this procedure. This is possible due to Lemma 8.

**Theorem 6.** *The pair $(L, R)$ produced by Algorithm* **NiceOrderedPartition** *defines a nice ordered partition. The running time of the algorithm is $O(n^3)$.*

*Proof.* (Hints.) The algorithm takes the one-blocks of $\mathcal{O}$ sorted in a topological order, and processes them one by one. Our goal is to obtain a pair of lists $(L, R)$ like in Theorem 5. Initially all the blocks are marked "$L$ or $R$", which means that they can be put either in $L$ or in $R$. When we decide to put a block in one of the lists, other blocks may be forced to appear in the opposite or in the same list, because of incomparability relations. Indeed if two blocks are incomparable with respect to $\preceq$, they must be placed on different sides of $B$. As described in the full proof, the graph $(\mathcal{O}(B), \|)$, defined by the relation $O \parallel O'$ iff $O$ and $O'$ are incomparable w.r.t. $\preceq$, is a bipartite graph. This forcing is explored through the procedure "shake", by marking the blocks "$L$" or "$R$". It corresponds to a breadth-first search of the graph $(\mathcal{O}(B), \|)$. Another key point in the proof of the algorithm is that the blocks of each connected component of this incomparability graph induce an interval in $\leq_{top}$.

We can think of the algorithm as having three phases. The first phase lasts as long as we do not encounter any hit block. The blocks considered in the first phase can be put in any convenient list; they will stay clean. When we meet the first hit block we enter the second phase. We denote by $MinHitSide$ the list containing this block. Suppose without loss of generality that this list is $L$. We are in the second phase as long as $R$ has no hit block. When we treat a block $O$ we should put it in $R$ if it is sparable; by putting it in $L$ it would be filled. On the contrary, if $O$ is not sparable we put it in $L$, thus keeping the possibility to spare other blocks. Eventually, the third phase starts when both $L$ and $R$ contain hit blocks. In this case we put the current block in any convenient list, it will be filled anyway. $\square$

## 5 Putting everything together

Given an interval graph $G = (V, E)$ and a vertex $x$ outside of $G$, we run the following procedure with $(G, x, N_{G'}(x))$ as input. A minimal interval completion of $G'$ is obtained as $H' = (V', E')$ with $V' = V \cup \{x\}$ and $E' = E \cup \{\{x, y\} \mid y \in \textbf{IntervalCompletion}(G, x, N_{G'}(x))\}$.

We construct the block $B$ as in Theorem 3 and then call the algorithm **NiceOrdered-Partition** to obtain the nice ordered partition $(L, R)$. In order to compute a minimal interval completion of $G'$, according to Lemma 7 it remains to compute, by recursive calls, minimal interval completions of $G'_d[O \cup \{x\}]$ for each $O \in \{O_L, O_R\}$.

Here we might encounter a difficulty that we have not yet discussed. During this recursive call on $G'_d[O \cup \{x\}]$ it is possible that $\mathcal{S}_x$ is restricted to the minimal separator $S$ bordering $O$ in

```
procedure NiceOrderedPartition
Input: (𝒪(B), ⪯, ≤_top) the pre-order on the set of all one-blocks together with a topological order of the refined pre-order
Output: (L, R) - lists of one-blocks to be constructed s.t. (L, R) defines a nice ordered partition
Variables: M - an array used to store forcing information on the one-blocks.
```

$$M[O] = \begin{cases} \text{“}L\text{”} & \text{- } O \text{ is forced to be in } L; \\ \text{“}R\text{”} & \text{- } O \text{ is forced to be in } R; \\ \text{“}L \text{ or } R\text{”} & \text{- O can be either in } L \text{ or in } R. \end{cases}$$

$MinHitFound$ - information if the minimal hit one-block has been found
$MinHitSide$ - information on the side where the minimal hit one-block has been put, initialized to be $L$ but $R$ would be fine as well

```
procedure shake(O)
forall X ∈ 𝒪 s.t (O ≤_top X) and (X is incomparable with O for ⪯) and (M[X] = “L or R”)
        M[X] := opposite(M[O]) do
        shake(X)
end

MinHitFound := false
MinHitSide := L
L ← ∅; R ← ∅
forall O ∈ 𝒪 do
        M[O] := “L or R”
forall O ∈ 𝒪 in the topological order do
        if (M[O] = “L or R”) then
           if O is sparable then
                move O on the top of opposite(MinHitSide) and M[O] := opposite(MinHitSide)
                shake(O)
           else
                move O on the top of MinHitSide and M[O] := MinHitSide
                shake(O)
           endif
        else
           move O on the top of M[O]
        endif
        if (not MinHitFound and O is hit) then
           MinHitFound := true
           MinHitSide := M[O]
        endif
endforall
return (L, R)
```

**Table 1.** Main case: the algorithm constructing a nice ordered partition $(L, R)$

$G$ (even when $G'_d[O \cup \{x\}]$ is not an interval graph). If we choose $S$ as the block $B$, the algorithm calls itself again on $G'_d[O \cup \{x\}]$ and it will not terminate. In this case we define $B$ as the union of maximal cliques of the input graph containing $N(x) \setminus \{d\}$, the neighborhood of $x$ except the dummy vertex. In this case also **NiceOrderedPartition** produces a nice ordered partition, as explained in Appendix C.

```
procedure MinIntervalCompletion
input: G, x, N - the neighborhood of x in G', d - dummy vertex
output: modified N - the neighborhood of x in an interval completion of G'

if G' is an interval graph then
   return ∅

Compute S_x.
if N(d) is not the only minimal separator in S_x then
         B := a maximal piece between two minimal separators in S_x
              or a minimal separator S ∈ S_x if such a piece between does not exist
         N := N ∪ B
else
         B := the union of all maximal cliques containing N \ {d}
endif
Compute the one-blocks O(B)
Check sparability of all one-blocks.
Compute the graph (O(B), ≤) and a topological order ≤_top
(L, R) ← NiceOrderedPartition((O(B), ≤, ≤_top))
forall O in L above O_L do N := N ∪ O
forall O in R above O_R do N := N ∪ O // no iterations if O_R is undefined
foreach O ∈ {O_L, O_R} do
   if O is sparable then
      let G_d[O] := G[O] plus a dummy vertex d adjacent to O ∩ B
      N := N ∪ MinIntervalCompletion(G_d[O], x, N ∩ O ∪ {d},d) \ {d}
   else if O is not sparable then N := N ∪ O
return N
```

**Theorem 7.** *Algorithm* **MinIntervalCompletion** *computes a minimal interval completion of $G'$ in $O(n^4)$ time.*

*Proof.* We show that one call of the procedure costs $O(n^3)$ time. The total numbers of calls is $O(n)$. Indeed if we consider, for each recursive call on $G_d[O]$, the separator bordering $O$, we notice that a separator is used at most twice. The number of minimal separators of $G$ is $O(n)$.

Computing the $O(n)$ minimal separators can be done in $O(n^2)$ time. Checking the sparability of all blocks can be done in $O(n^3)$ time using Lemma 8. Indeed for each one-block $O$ and for each vertex $y$ of $O$ like in Lemma 8 we check if the graph obtained from $G'_d[O \cup \{x\}]$ by filling $O \setminus y$ is an interval graph. The latter graph has $O(n)$ vertices. The total number of vertices $y$ that we have to consider is at most $n$, because in different blocks we take different vertices. Consequently the sparability of all blocks can be checked in $O(n^3)$ time.

Computing the graph $(\mathcal{O}(B), \leq)$ can be done in $O(n^3)$ time. □

**Theorem 8.** *There is an algorithm computing a minimal interval completion of an arbitrary graph in $O(n^5)$ time.*

*Proof.* Let $\{x_1, \dots, x_n\}$ be an arbitrary ordering of the vertices of $G$. We compute incrementally a minimal interval completion $H_i$ of $G_i = G[\{x_1, \dots, x_i\}]$, respecting $H_{i-1}$. For this purpose we use Theorem 11 (Appendix $A$) if the neighborhood of $x_i$ in $G_{i-1}$ induces a clique in $H_{i-1}$ and Theorem 7 otherwise. □

## 6 Conclusion

In this paper we give the first polynomial time algorithm that computes a minimal interval completion of an arbitrary graph. With small modifications, our algorithm can be transformed to produce any of the minimal interval completions of the input graph. A natural question is whether the algorithm can be adapted into heuristics specialized in finding interval completions with few fill edges or small clique size. For example, at each iteration we are able to choose an interval completion that (locally) adds a minimum number of edges.


## Acknowledgment

The authors would like to thank Fedor Fomin for useful discussions on the subject.


## References


1. A. Berry, P. Heggernes, and Y. Villanger. A vertex incremental approach for dynamically maintaining chordal graphs. In *Algorithms and Computation - ISAAC 2003*, pages 47–57. Springer Verlag, 2003. LNCS 2906.
2. V. Bouchitté and I. Todinca. Treewidth and minimum fill-in: grouping the minimal separators. *SIAM J. on Computing*, 31(1):212 – 232, 2001.
3. M. R. Garey and D. S. Johnson. *Computers and Intractability.* W. H. Freeman and Co., 1978.
4. P. C. Gilmore and A. J. Hoffman. A characterization of comparability graphs and of interval graphs. *Canadian J. Math.*, 16:539–548, 1964.
5. M. C. Golumbic. *Algorithmic Graph Theory and Perfect Graphs.* Academic Press, 1980.
6. P. W. Goldberg, M. C. Golumbic, H. Kaplan, and R. Shamir. Four strikes against physical mapping of DNA. *J. Comput. Bio.*, 2(1):139–152, 1995.
7. J. Gustedt. On the pathwidth of chordal graphs. *Discrete Applied Mathematics*, 45(3):233–248, 2003.
8. P. Heggernes. Minimal triangulations of graphs: A survey. To appear *Discrete Math.*
9. T. Kloks, D. Kratsch, and J. Spinrad. On treewidth and minimum fill-in of asteroidal triple-free graphs. *Theor. Comput. Sci.*, 175:309–335, 1997.
10. C.G. Lekkerkerker and J.C. Boland. Representation of a finite graph by a set of intervals on the real line. *Fund. Math.*, 51:45–64, 1962.
11. A. Parra and P. Scheffler. Characterizations and algorithmic applications of chordal graph embeddings. *Disc. Appl. Math.*, 79:171–188, 1997.
12. D. Rose, R.E. Tarjan, and G. Lueker. Algorithmic aspects of vertex elimination on graphs. *SIAM J. Comput.*, 5:146–160, 1976.
13. M. Yannakakis. Computing the minimum fill-in is NP-complete. *SIAM J. Alg. Disc. Meth.*, 2:77–79, 1981.

## A Proof of Theorem 6

The proof of Theorem 6 will be established through the following series of results.

### A.1 The co-comparability graph

Let us define the *parallel* relation $\|$ such that $O_1 \| O_2$ if and only if $O_1 \not\preceq O_2$ and $O_2 \not\preceq O_1$. Let $(\mathcal{O}(B), \|)$ be the graph on vertex set $\mathcal{O}(B)$ in which two vertices are adjacent if and only if they are parallel. This graph is the co-comparability graph of $(\mathcal{O}(B), \preceq)$; furthermore, it is also the co-comparability graph of $(\mathcal{O}(B), \leq)$.

**Lemma 9.** *Graph $(\mathcal{O}(B), \|)$ is bipartite.*

*Let $O_1, O_2 \in \mathcal{O}$ which are forced to be in the same color class of $(\mathcal{O}(B), \|)$. For any clique path of $G$, the two one-blocks are on a same side of $B$.*

*If $O_1, O_2$ are forced to be in different color classes, then they are separated by $B$ in any clique path of the graph.*

*Proof.* According to Lemma 6, if two blocks $O$ and $O'$ are incomparable with respect to the $\preceq$ relation, they must be separated by $B$ in any clique path $CP_G$ of $G$. Thus, by assigning color "$L$" to all one-blocks to the left of $B$ in $CP_G$, and color "$R$" to all those to its right, we obtain a two-coloring of $(\mathcal{O}(B), \|)$.

The next observations come directly from the fact that if $O$ and $O'$ must be in the same color class (resp. in different color classes), then there is a path of even (resp. odd) length from $O$ to $O'$ in $(\mathcal{O}(B), \|)$. We then use the fact that two incomparable blocks must be on different sides of $B$. ☐

Because of these properties, sometimes we will have to make an arbitrary choice and spare only one of two one-blocks. Suppose we have already established minimal hit one-blocks in both $L$ and $R$. Let $O_1 \| O_2$ and $O_L \leq O_1, O_2 \leq O_R$. Thus we can spare only one of them and any choice will be correct.

**Lemma 10.** *Let $C$ and $D$ be two distinct connected components of $(\mathcal{O}(B), \|)$. There is no triple of one-blocks $O_c, O'_c, O_d$ such that $O_c, O'_c \in C$, $O_d \in D$ and $O_c \preceq O_d \preceq O'_c$.*

*Proof.* Suppose that such a triple exists and consider a path $P$ joining $O_c$ and $O'_c$ in $(\mathcal{O}(B), \|)$: $P = [O_c = O_1, O_2, \ldots, O_p = O'_c]$. Then there are two consecutive one-blocks $O_i$ and $O_{i+1}$ on this path such that $O_i \preceq O_d \preceq O_{i+1}$. By the transitivity of the $\preceq$ relation, $O_i \preceq O_{i+1}$, contradicting the fact that the two one-blocks are adjacent in $(\mathcal{O}(B), \|)$. ☐

**Definition 7.** *Let us extend the order $\leq$ to the set of connected components of $(\mathcal{O}(B), \|)$. Define the order $(\mathcal{C}(B), \leq)$ with $\mathcal{C}$ being the set of connected components of $(\mathcal{O}(B), \|)$ and $C \leq D$, $C, D \in \mathcal{C}$ if $O_c \leq O_d$ for all $(O_c, O_d) \in (C, D)$.*

By Lemma 10 we deduce directly :

**Lemma 11.** *For any two connected components $C$ and $D$ of the graph $(\mathcal{O}(B), \|)$, we have $C \leq D$ or $D \leq C$. Moreover, if there are two blocks $O_c \in C$ and $O_d \in D$ that are equivalent with respect to $\leq$, then each of the components is composed by a single block.*

Consequently, the topological order used by the algorithm induces an order $C_1, \ldots, C_p$ on the connected components of $(\mathcal{O}(B), \|)$. Moreover, the algorithm does not encounter blocks of $C_i$ before it has treated all the blocks of $C_1, \ldots, C_{i-1}$.

### A.2 The proof of correctness of Algorithm NiceOrderedPartition

Notice that choosing the topological order and coloring the graph $(\mathcal{O}(B), \|)$ explores all the freedom we have for creating a nice ordered partition. We will first prove that the topological order chosen is without impact for the nice property of an ordered partition and then that our algorithm gives a nice ordered partition.

**Lemma 12.** *Let $\mathcal{O}_{eq}$ be a set of one-blocks equivalent for $\leq$. Then they are either in separate singleton connected components of $(\mathcal{O}(B), \|)$ or they are forced to be in the same color class of one connected component of $(\mathcal{O}(B), \|)$.*

*Proof.* Let $O_{eq} \in \mathcal{O}_{eq}$. If there is $O \parallel O_{eq}$ in $\mathcal{O}(B)$ then $O$ is incomparable with the rest of one-blocks in $\mathcal{O}_{eq}$. Thus all $\mathcal{O}_{eq}$ are in the same connected component of $(\mathcal{O}(B), \|)$, in the opposite color class than $O$. Else, there is no $O \parallel O_{eq}$. Hence $\mathcal{O}_{eq}$ form singleton connected components of $(\mathcal{O}(B), \|)$. □

**Lemma 13.** *Two one-blocks in a maximal set of one-blocks $\mathcal{O}_{eq}$ equivalent for $\leq$ cannot be separated by a one-block not in $\mathcal{O}_{eq}$.*

*Proof.* Suppose $O_1, O_2 \in \mathcal{O}_{eq}$ and there is $O \notin \mathcal{O}_{eq}$ such that $O_1 \leq O \leq O_2$. By equivalence we have that $O_2 \leq O_1$ so $O_1 \leq O \leq O_1$. Thus $O \in \mathcal{O}_{eq}$ - a contradiction. □

**Theorem 9.** *For any topological order $\leq_{top}$ there is a nice ordered partition $(L_{\leq_{top}}, R_{\leq_{top}})$ respecting it, in the sense that the the two lists are suborders of $\leq_{top}$.*

*Proof.* Let $(L, R)$ be a nice ordered partition corresponding to a topological order. Permute two elements $O_1$ and $O_2$ in this topological order. Let us construct a nice ordered partition $(L', R')$ respecting the new topological order.

If $O_1$ and $O_2$ are incomparable w.r.t. $\leq$, then $(L', R') = (L, R)$.

Let $\mathcal{O}_{eq}$ be a maximal set of one-blocks equivalent for $\leq$, containing $O_1$, $O_2$. Then $L'$ and $R'$ are obtained from $(L, R)$ by exchanging $O_1$ and $O_2$ (they may be in the same list or in different lists). Suppose that $(L', R')$ is not nice. That means there is a one-block $O$ that can be put outside the interval $O_{L'}$-$O_{R'}$ and spared.
Suppose $O \notin \mathcal{O}_{eq}$. Then $O$ can be put outside the corresponding interval in $(L, R)$ too. Indeed, the constraints coming from $\leq$ are the same, no matter what is the permutation of one-blocks $\mathcal{O}_{eq}$. Moreover, by Lemma 12 and Lemma 13 the placement of one-blocks $O \notin \mathcal{O}_{eq}$ relative to $O_L$ and $O_R$ is the same in $(L, R)$ and in $(L', R')$.
Suppose $O$ belongs to $\mathcal{O}_{eq}$. If all $\mathcal{O}_{eq}$ belong to the same connected component of $(\mathcal{O}(B), \|)$ then they form an interval of one-blocks at the same side of $B$ in every ordered partition. Say that there are in $L$. If $O_L$ is bigger or smaller than $\mathcal{O}_{eq}$ then a transposition does not change the set $\mathit{Spared}(L, R)$. So $O_L \in \mathcal{O}_{eq}$ and all elements of $\mathcal{O}_{eq}$ are hit but sparable. Since only one one-block of $\mathcal{O}_{eq}$ can be spared in such setting, we have that $O = O_{L'}$ - a contradiction.
So the one-blocks of $\mathcal{O}_{eq}$ form singleton connected components of $(\mathcal{O}(B), \|)$. The number of one-blocks among $\mathcal{O}_{eq}$ that can be spared depends on the number of sides with hit one-blocks smaller than $O_{eq}$ in $(L, R)$. This maximum is achieved in $(L, R)$, thus in $(L', R')$ too. If $O_{eq}$ is clean (then all or no one-block in $\mathcal{O}_{eq}$ is in $\mathit{Spared}(L, R)$) or $O_{eq}$ is hit but sparable then $O$ is already spared in interval completions good $(L', R')$. If $O_{eq}$ is not sparable then $O$ cannot be spared. Anyway, we reach a contradiction. □

In the proof we will show that at any step of the algorithm we obtain a "partial" nice ordered partition $(L, R)$. It is characterized by Theorem 5, only that the set of one-blocks considered is restricted to the ones already partitioned into $(L, R)$.

**Theorem 10.** *Given graph $(\mathcal{O}(B), \leq)$ for $G$ and $B$, the algorithm creates a nice ordered partition $(L, R)$.*

*Proof.* Let us see the algorithm in terms of connected components $C_1, \ldots, C_k$ of $(\mathcal{O}(B), \|)$. They are processed in topological order coming from the topological order of one-blocks in $\mathcal{O}(B)$. Each time the algorithm finds a one-block with a label "L or R", it is the minimal one-block of a new component encountered. The choice of putting it in one side decides, through forcing, about the partitioning of the whole component. Thus $(L, R)$ can be seen as a function assigning to each connected component of $(\mathcal{O}(B), \|)$ a binary value 1 if the first color class is put into $L$ and 0 - otherwise. Let us call it a *binary representation.* Let $\mathcal{O}_1(C_i), \mathcal{O}_2(C_i)$ denote the two color classes of $C_i$ and suppose w.l.o.g. that the minimum element of $C_i$, with respect to $\leq_{top}$, is in $\mathcal{O}_1(C_i)$.

We prove that at every step the algorithm creates $(L_i, R_i)$ from $(L_{i-1}, R_{i-1})$ by choosing to add one color class of $C_i$ to $L_{i-1}$ and the second to $R_{i-1}$ in such a way that the set $Spared(L_i, R_i) \setminus Spared(L_{i-1}, R_{i-1})$ is inclusion maximal among the two possible choices. Creating $(L, R)$ can be divided into three phases. During the first phase, none of the lists contains hit blocks. For the second phase only one list has blocks, and in the third phase both lists have hit blocks.

1. As long as the components $C_1, \ldots, C_{i-1}$ do not contain any hit one-block, we add $\mathcal{O}_1(C_i)$ to $L$ and $\mathcal{O}_2(C_i)$ to $R$. Clearly, this decision spares the same set of one-blocks as the other, thus we have maximality. This stage stops when we encounter $C_i$ with a hit one-block. Then we fix the `MinHitSide` to point the side containing the first encountered hit one-block, w.l.o.g. assume that "$L$" is the side and $O_L$ is the one-block. If "$R$" does not contain a hit one-block then we pass to the step 2, which lasts as long as only "$L$"contains hit blocks. If "$R$" contains a hit one-block mark $O_R$ and pass to the step 3, when both lists contain hit blocks.
2. Recall that at this stage only the left list contains hit blocks. For each $C_i$ we add $\mathcal{O}_1(C_i)$ to $R$ if and only if its minimum element $O_i$ is sparable. If $O_i$ is sparable, our choice ensures that $O_i \in Spared(L_i, R_i)$, while the other choice would imply $O_i \notin Spared(L_i, R_i)$. If $O_i$ is not sparable, by our choice we spare the minimum element $O'_i$ in the opposite color class of $C_i$ (if the latter is sparable). If $O'_i$ is not sparable, either choices would imply $Spared(L_i, R_i) = Spared(L_{i-1}, R_{i-1})$. We have maximality again. This stage stops when we put a hit one-block in "$R$" and pass to the step 3.
3. We just add $\mathcal{O}_1(C_i)$ to $L$ and $\mathcal{O}_2(C_i)$ to $R$. Whatever the decision is, no more one-blocks can be spared in this step.

So at each step the algorithm chooses an inclusion maximal augmentation of the set of spared one-blocks. Now let $(L, R)$ be the ordered partition obtained by the algorithm. Suppose there is $(L', R')$ with $Spared(L, R) \subset Spared(L', R')$. $(L', R')$ can not be obtained from $(L, R)$ by permutation of elements without changing the sides. As in the proof of Theorem 9, permutations of elements equivalent w.r.t. the refined preorder does not increase the set *Spared*. For the sake of simplicity of reasoning but w.l.o.g., we can assume that there are no components equivalent for $\leq$, thus the topological ordering of components is unique.

Take $(L', R')$ to be one sharing with $(L, R)$ the longest prefix of the binary representation among the ordered partitions having the properties described above. Let $C$ be the first, in topological order, component partitioned in $(L', R')$ in another way than in $(L, R)$. We can assume that $C$ is also the fist component that contains more spared one-blocks in $(L', R')$ than in $(L, R)$ - otherwise we could partition this and all bigger components the other way obtaining $(L'', R'')$ with $Spared(L'', R'') = Spared(L', R')$ and sharing longer prefix with $(L, R)$. That means that for $C$ the algorithm did not choose the twist adding an inclusion maximal set of one-blocks to the set of spared ones. A contradiction. $\square$

### A.3 The running time of Algorithm NiceOrderedPartition

The **shake** procedure can be implemented as a depth-first search in the graph $(\mathcal{O}(B), \|)$. This graph has at most $n$ vertices and $O(n^2)$ edges, thus the overall running time of the procedure is $O(n^2)$. The graph $(\mathcal{O}(B), \|)$ can be constructed in $O(n^3)$ time. The rest of the algorithm only requires $O(n)$ time.

## B Special case: the neighborhood of $x$ is a clique

Assume that $G'$ is not an interval graph. Recall that, by Lemma 1, each minimal separator of $G$ is the intersection of two maximal cliques that are adjacent in a clique path.

**Lemma 14.** *Let $\mathcal{K}_{max}$ be the set of maximal cliques containing $N(x)$. Consider a clique path $CP_G$ of $G$ and let $S = K \cap K'$ be a minimal separator of $G$, where $K$ and $K'$ are maximal cliques incident in $CP_G$ and $K \in \mathcal{K}_{max}$. The graph $H$ obtained from $G'$ by filling $S$ is an interval completion of $G'$.*

*Proof.* Splitting edge between $K$ and $K'$, add a new bag $S \cup N[x]$ between $K$ and $K'$. Since $N(x) \subset K$ this yields a clique path of $H$. ◻

We prove that any minimal interval completion of $G$ is obtained in this way.

**Lemma 15.** *Let $H$ be a minimal interval completion of $G$ and $CP_G$ be a clique path of $G$ obtained by pruning the vertex $x$ from a clique path $CP_H$ of $H$. Let $\mathcal{K}_{max}$ be the set of maximal cliques containing $N(x)$. There is a minimal separator $S$ of $G$ corresponding to an edge of $CP_G$ incident to $\mathcal{K}_{max}$ such that $S$ is filled in $H$.*

*Proof.* Notice that in $H$ there is only one maximal clique $\Omega$ containing $x$. Otherwise we could pick one maximal clique of the subpath containing $N_{G'}[x]$ and remove $x$ from the others, with the result still being an interval supergraph of $G'$. Maximal clique $\Omega$ cannot be placed at any end of $CP_H$, since it would mean that $G'$ is already an interval graph, with $\Omega = N_{G'}[x]$. Let $K$ and $K'$ be the maximal clique of $H$, adjacent to $\Omega$ in $CP_H$. Let $K''$ be a maximal clique of $G$ containing $N_{G'}(x)$. If $K \subset \Omega$, then the result follows by taking any separator $S$ corresponding to an edge of $CP_G$, incident to $K''$. Otherwise at least one of $K$ and $K'$ (say $K$) contains $N_{G'}(x)$. Thus $K \in \mathcal{K}_{max}$. The maximal cliques $K$ and $K'$ of $G$ are adjacent in $CP_G$ and the minimal separator $S = K \cap K'$ is filled in $H$. ◻

**Theorem 11.** *There is a minimal separator $S$ of $G$ such that filling it in $G'$ yields a minimal interval completion of $G'$. This minimal interval completion can be found in $O(nm)$ time.*

*Proof.* Let $H$ be a minimal interval completion of $G'$. By the previous lemma, there exists a clique path $CP_G$ of $G$ such that one of the separators $S$ corresponding to an edge of $CP_G$ incident to $\mathcal{K}_{max}$ is filled in $H$. By Lemma 14, it is sufficient to fill $S$ in order to obtain an interval completion of $G'$, so $H$ is exactly the graph obtained from $G'$ by filling $S$.

In order to obtain the minimal completion $H$, we consider each minimal separator $S$ of $G$ and check whether the graph obtained by filling it is an interval graph. Then it remains to choose a minimal one. In an interval graph there are at most $n-1$ minimal separators computable in linear time. For each of them, checking whether the filled graph is interval takes a linear time. Choosing an inclusion minimal one can be achieved in $O(n^2)$ time. ◻

## C Special case: $\mathcal{S}_x$ is reduced to the $N_G(d)$

We are in the situation when the algorithm **MinIntComp** was called on a graph $G$ having a dummy vertex $d$ and the only minimal $a,b$-separator of $G$ with $a, b \in N_{G'}(x)$ is $N_G(d)$. Recall that the vertices of $G - d$ correspond to a one-block of the input graph of the procedure **MinIntComp** at the higher level of recursion. Let $O_G = S_G \cup C_G$ be this one-block, where $S_G = N_G(d)$ is the minimal separator bordering it.

We have defined $B$ as the union of all maximal cliques of $G$ containing $N_{G'}(x) \setminus \{d\}$. Hence the maximal cliques contained in $B$ form a subpath in any clique path of $G$. Recall that $\mathcal{O}(B)$ is the set of one-blocks of the type $S \cup C$, where $C$ is a component of $G - B$ and $S = N_G(C)$.

**Lemma 16.** *For any block $O \in \mathcal{O}(B)$, $x$ has a neighbor in $B \setminus O$.*

*Proof.* Assume, on the contrary, that $N_{G'}(x) \setminus \{d\}$ is contained in $O$. Since this set induces a clique, it will be contained in some maximal clique of $G[O]$, contradicting the construction of $B$. □

**Lemma 17.** *The clique $K_d = N_G[d]$ is one of the one-blocks of $\mathcal{O}(B)$.*

*Proof.* Let $C_d$ be the component of $G - B$ containing $d$ and let $S_d = N_G(C_d)$. By Lemma 16, there exists a vertex $a \in B \setminus (S_d \cup C_d)$. Clearly $S_d$ is an $a,d$-separator. Since $B$ is a block, $S_d$ is actually a minimal $a,d$-separator. By the fact that $a, d$ are neighbors of $x$ and $\mathcal{S}_x = \{S_G\}$ we conclude that $S_d = S_G$ and $C_d = \{d\}$. □

Theorem 4 applies also in this case, but Lemma 7 has to be slightly modified. Observe that $K_d$ is the only one-block of $\mathcal{O}(B)$ which is not clean. Clearly, it is not sparable. Let $(L, R)$ define a valid ordered partition induced by $\mathcal{O}(B) \cup \{B\}$. Assume without loss of generality that $K_d \in L$, hence $O_L = K_d$ and $R$ contains no hit block. Here again we say that an interval completion $H$ of $G'$ is good for $(L, R)$ if it is inclusion-minimal among the interval completions respecting this valid ordered partition. The lemma characterizing good interval completions for $(L, R)$ becomes:

**Lemma 18.** *A supergraph $H$ of $G'$ is a good interval completion for $(L, R)$ if and only if:*

1. *For each block $O$ appearing after $O_L = K_d$ in $L$, $O$ is filled in $H$.*
2. *The blocks appearing before $K_d$ in $L$ and the blocks of $R$ stay clean in $H$.*

*Proof.* Suppose that $H$ is an interval completion of $G'$ respecting some clique path $CP_G$ of $G$, obtained by refining $(L, R)$. Consider a one-block $O$ appearing after $K_d$ in $L$, so $P_O$ is between $P_{K_d}$ and $P_B$ in the clique path. Note that $x$ has a neighbor in $K_d \setminus O$, namely the dummy vertex $d$. By Lemma 16, it also has some neighbor in $B \setminus O$. Therefore $x$ must appear in all the maximal cliques contained in $O$. Hence $O$ is filled.

Conversely, consider the graph $H'$ obtained from $G'$ by filling all the blocks $O$ appearing after $K_d$ in $L$, we prove that $H'$ is an interval graph. From any clique path $CP_G = (\mathcal{K}, P)$ of $G$, we construct a clique path $CP_{H'}$ of $H'$ as follows. Add $x$ to $K_d$ and all the cliques contained in some block appearing after $K_d$ in $L$. If $N_{G'}(x) \setminus \{d\}$ is already a maximal clique of $G$, then $B$ is reduced to this clique and we simply add $x$ to it. Otherwise let $e$ be the edge of the path $P$ appearing between $P_B$ and the subpath corresponding to the last block of $L$. We denote by $S_L$ the minimal separator of $G$ corresponding to the edge $e$ (in the sense of Lemma 1). Add on this edge a new node whose corresponding clique is $x \cup S_L \cup (N_{G'}(x) \setminus \{d\})$. It is not hard to check that we obtained a clique path of $H'$. We conclude that $H'$ is a good interval completion for $(L, R)$. □

By Lemma 18, here again we deduce that a valid ordered partition $(L, R)$ is nice if and only $Spared(L, R)$ is inclusion-maximal like in Theorem 5. The ordered partition produced by the algorithm **NiceOrderedPartition** insures this maximality, therefore it is nice. Moreover we shall have $O_L = K_d$ and $O_R$ is undefined, thus the algorithm **MinIntComp** produces (in one step) a minimal interval completion of $G$.