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 completion through graph exploration

Karol SUCHAN
LIFO, Université d'Orléans
Ioan TODINCA
LIFO, Université d'Orléans

Rapport N^o **2006-08**

Minimal Interval Completion Through Graph Exploration

Karol Suchan[1,2] and Ioan Todinca[1]

[1] LIFO, Université d'Orléans, 45067 Orléans Cedex 2, France,
`Karol.Suchan,Ioan.Todinca@univ-orleans.fr`
[2] Department of Discrete Mathematics, Faculty of Applied Mathematics,
AGH - University of Science and Technology, Cracow, Poland

July 11, 2006

Abstract. Given an arbitrary graph $G = (V, E)$ and an interval graph $H = (V, F)$ with $E \subseteq F$ we say that H is an *interval completion* of G. The graph H is called a *minimal interval completion* of G if, for any sandwich graph $H' = (V, F')$ with $E \subseteq F' \subset F$, H' is not an interval graph. In this paper we give a $\mathcal{O}(nm)$ time algorithm computing a minimal interval completion of an arbitrary graph. The output is an interval model of the completion.

1 Introduction

Various well-known graph parameters, like *treewidth*, *minimum fill-in*, *pathwidth* or *bandwidth* are defined in terms of graph embeddings. The general framework consists in taking an arbitrary graph $G = (V, E)$ and adding edges to G in order to obtain a graph $H = (V, E \cup E')$ belonging to a specified class $\mathcal{H}$. For example, if H is chordal then it is called a *triangulation* of G. The *treewidth* can be defined as $\min(\omega(H)) - 1$, where the minimum is taken over all triangulations of G (here $\omega(H)$ denotes the maximum size of a clique in H). If, instead of minimizing $\omega(H)$, we minimize $|E'|$, the number of added edges, we define the *minimum fill-in* of G. If $H = (V, E \cup E')$ is an interval graph, we say that H is an interval completion of G. The *pathwidth* of G can be defined as $\min\{\omega(H)) - 1 \mid H \text{ is an interval completion of } G\}$. The minimum number of edges that we need to add for obtaining an interval completion is called the *profile* of the graph.

For each of the parameters cited above, as well as for similar embedding problems into other type of graph classes, the problem of computing the parameter is NP-hard. Obviously, for all of them, the optimal solution can be found among the *minimal* embeddings. We say that $H = (V, E \cup E')$ is a *minimal triangulation* (*minimal interval completion*) if no proper subgraph of H is a triangulation (interval completion) of G.

Computing minimal triangulations is a standard technique used in heuristics for the treewidth or the minimum fill-in problem. The deep understanding of minimal triangulations lead to many theoretical and practical results for the treewidth and the minimum fill-in. We believe that, similarily, the study of other types of minimal completions might bring new powerfull tools for the corresponding problems.

Related work. Much research has been devoted to the minimal triangulation problem. Tarjan and Leuker propose the first algorithm solving the problem in $O(nm)$ time. Several authors give different approaches for the same problem, with the same running time. Only recently this $O(nm)$ (in the worst case $O(n^3)$) time complexity has been improved by the algorithms of Kratsch and Spinrad ([13], running in $\mathcal{O}(n^{2.69})$ time) and Heggernes, Telle and Villanger ([12], running in $\mathcal{O}(n^\alpha \log n)$ time where $\mathcal{O}(n^\alpha)$ is the time needed for the multiplication of two

$n \times n$ matrices). The latter algorithm is the fastest up to now for the minimal triangulation problem.

A first polynomial algorithm solving the minimal interval completion problem is given in [11], using an incremental approach. Recent results relate to minimal completions into split and comparability graphs [9, 10].

Our result. We study the minimal interval completion problem. Our main result is an $\mathcal{O}(nm)$ time algorithm computing a minimal interval completion of an arbitrary graph, faster and simpler than the result of [11]. The latter result is based on characterization of interval graph by existence of its clique path. Here, we use the characterization by a special ordering of the vertex set, called interval ordering [17]. Its role is similar to the simplicial elimination scheme for chordal graphs. We define a family of orderings such that the associated proper interval graph is a minimal interval completion. Eventually, we give an $\mathcal{O}(nm)$ time algorithm computing such an ordering. Our algorithm is based on a breadth-first search of the input graphs, using special tie-break rules. In particular, we use the LexBFS algorithm for tie-breaks. The ordering can be efficiently transformed into an interval model.

2 Definitions and basic results

Let $G = (V, E)$ be a finite, undirected and simple graph. Moreover we only consider connected graphs — in the non-connected case each connected component can be treated separately. Denote $n = |V|$, $m = |E|$. If $G' = (V', E')$ is a spanning subgraph of $G = (V, E)$ (i.e. $V' = V$ and $E \subseteq E'$) we write $G \subseteq G'$ (and $G \subset G'$ if $G \subseteq G', G \neq G'$). The *neighborhood* of a vertex v in G is $N_G(v) = \{u \mid \{u, v\} \in E\}$. Similarly, for a set $A \subseteq V$, $N_G(A) = \bigcup_{v \in A} N_G(v) \setminus A$. The *closed neighborhood* of A (of v) is $N_G[A] = A \cup N_G(A)$ ($N_G[v] = \{v\} \cup N_G(v)$). As usual, the subscript is sometimes omitted.

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. The family of intervals is called the *interval model* of the graph.

Theorem 1 ([5]). *A graph G is interval if and only if there is a path P whose vertex set is the set of all maximal cliques of G, such that the subgraph of 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* of G. Notice, that a clique path P gives an interval model of G, with an interval (subpath) of maximal cliques assigned to each vertex. For our purpose, we also use the caracterization of interval graphs in terms of vertex orderings (also called layouts).

Definition 1 (interval ordering [17]). *An* interval ordering *of the vertices of a graph $H = (V, F)$ is a linear ordering $\sigma = (v_1, v_2, \ldots, v_n)$ of V such that, for any $1 \leq i < j \leq k \leq n$, if $\{v_i, v_k\} \in F$ then also $\{v_i, v_j\} \in F$.*

Theorem 2 ([17]). *A graph $H = (V, F)$ is an interval graph if and only if there exists an interval ordering of its vertex set.*

Definition 2. *Let $G = (V, E)$ be an arbitrary graph and $\sigma = (v_1, \ldots, v_n)$ be an ordering of V. The graph $G(\sigma) = (V, F)$ is defined by*

$$F = \{\{v_i, v_k\} \mid \text{ there is } j \text{ such that } 1 \leq i < k \leq j \leq n \text{ and } \{v_i, v_j\} \in E\}.$$

The following Lemma is a direct consequence of Theorem 2.

Lemma 1. *$G(\sigma)$ is an interval graph.*

Remark 1. Let $\sigma = (v_1, v_2 \dots, v_n)$ be an interval ordering of an interval graph H. An interval model of H can be obtained by associating to each vertex v_i the interval $[i, j]$, where $j \geq i$ is the largest index such that $\{v_i, v_j\} \in F$.

Conversely, given an interval model of the graph H, we obtain an interval ordering by ordering the vertices according to the left-end point of their intervals, from left to right. Ties can be broken arbitrarily. For technical reasons, in this article, we decide to use the right-ends as a tie-break, from left to right, too.

Given an interval model, a clique path can be obtained by traversing the model from left to right and, at each point p where an interval finishes, adding the clique of intervals intersecting p to the model if it is not included in the (maximal) clique added right before. If $H = G(\sigma)$, for some simple graph G, let $P(G, \sigma)$ denote the clique path obtained in that way.

Theorem 3. *Let $G = (V, E)$ be an arbitrary graph and $H = (V, F)$ be a minimal interval completion of G. Then there is an ordering σ such that $H = G(\sigma)$.*

Proof. By Theorem 2, there is an ordering σ of V such that $H = H(\sigma)$. As a straight consequence of Definition 2, $E(G(\sigma)) \subseteq E(H)$. By Lemma 1, $G(\sigma)$ is also an interval graph. Thus, by minimality of H, we deduce that $E(G(\sigma)) = E(H)$. □

Definition 3. *An ordering $\sigma = (v_1, \dots, v_n)$ is called* nice *if $G(\sigma)$ is a minimal interval completion of G. Any prefix $(v_1, \dots, v_k)$, $k \leq n$ of a nice ordering is called a* nice prefix.

Our goal will be to find a nice ordering σ of an arbitrary graph G. This will be achieved through ordered partitions of the vertex set, which are to be refined into a linear ordering.

Definition 4. *A tuple of disjoint subsets of V, $OP = (V_1, \dots, V_k)$ whose union is exactly V is called an* ordered partition *of V. A* refinement *of OP is an ordered partition OP' obtained by replacing each set V_i by an ordered partition of V_i.*

Definition 5. *Given an ordered partition $OP = (V_1, \dots, V_k)$, any tuple $OP' = (V_1, \dots, V_j)$, with $0 \leq j \leq k$, is called a* prefix *of OP. We use $\mathrm{V}(OP')$ to denote $\bigcup \{V_i \mid 1 \leq i \leq j\}$.*

In the particular case where $OP = (V_1)$, we simply write V_1. Moreover if V_1 is formed by a single vertex x, we write x instead of $\{x\}$. Given two tuples $OP' = (V_1, \dots, V_k)$, $OP'' = (V_{k+1}, \dots, V_{k+l})$, their concatenation $OP = (V_1, \dots, V_k, V_{k+1}, \dots, V_{k+l})$ is denoted by $OP' \bullet OP''$.

Notice that an ordering $\sigma = (v_1, \dots, v_n)$ of V is a special case of an ordered partition.

3 Nice orderings and nice prefixes

3.1 Choosing a first vertex

A *module* is a set of vertices M such that for any $x, y \in M$, $N(x) \setminus M = N(y) \setminus M$. A clique module is a module inducing a clique. An inclusion-maximal clique module will be simply called a *maximal clique-module.*

A *minimal separator* S is a set of vertices such that there exist two connected components of $G - S$ with vertex sets C and D satisfying $N(C) = N(D) = S$.

Lemma 2 (see e.g. [6]). *Let P be a clique path of an interval graph H. For any minimal separator S of H, there exist two maximal cliques of H, consecutive in P, whose intersection is S.*

Definition 6 ([1]). *A* moplex *is a maximal clique module M, such that $N(M)$ is a minimal separator of G. The vertices of a moplex are called* moplexian vertices*.*

The LexBFS (Lexicographic Breadth-First Search) algorithm, introduced by Rose, Leuker and Tarjan [19], is a famous linear-time algorithm that numbers the vertices of an arbitrary graph from n to 1. Initially designed to obtain a simplicial ordering for chordal graphs, we use it here to obtain the first vertex of a nice ordering. LexBFS is a particular breadth-first search algorithm (see Figure 1). Each vertex x has a label $lab(x)$, which is a tuple of integers. During the algorithm, each vertex x also receives a number. The algorithm may start the exploration of the graph on any vertex.

Algorithm `LexBFS`
Input: $G = (V, E)$ connected
Output: a numbering of the vertices from n to 1

// init
for each vertex x **do**
  x is marked "unnumbered"
  $lab(x) := \emptyset$
// main loop
for $i := n$ downto 1 **do**
  pick an unnumbered x with maximum label according to the lexicographic order
  give the number i to x
  for each unnumbered neighbour y of x **do**
    add the number i at the end of $lab(y)$

Fig. 1. The `LexBFS` algorithm.

Theorem 4 ([1]). *The algorithm LexBFS ends on a moplexian vertex.*

A vertex v numbered 1 by some execution of LexBFS is called a *LexBFS-terminal vertex*. A moplex M such that some execution of LexBFS terminates on a vertex of M is called a *LexBFS-terminal moplex*.

Lemma 3 ([1, 2]). *Let M be a LexBFS-terminal moplex and $S = N_G(M)$. Denote by $C_1, C_2, \ldots, C_k$, with $C_k = M$, the connected components of $G - S$ in the order in which the LexBFS execution encounters them. Then the following equation are satisfied:*

$$N(C_1) \subseteq N(C_2) \subseteq \cdots \subseteq N(C_k). \tag{1}$$

$$\begin{aligned} \forall i, j, x, y : 1 \leq i < j \leq k, x \in N(C_i), y \in N[C_j] \setminus N(C_i) \\ \Rightarrow \{x, y\} \in E(G). \end{aligned} \tag{2}$$

Lemma 4. *Consider a non-complete graph $G = (V, E)$. Let v be a vertex of a moplex M and $S = N_G(M)$. Let $C_1, C_2, \ldots, C_k$, with $C_k = M$, the connected components of $G - S$, satisfy Equations 1 and 2 of Lemma 3. Then there exists a minimal interval completion H of G such that $N_G(v) = N_H(v)$.*

For any such H, there exists a clique path P of H such that $M \cup S$ is one of its end cliques.

Proof. Let H' be the graph obtained from G by transforming $N_G[C_i]$ into a clique, from each $1 \leq i \leq q$. By Equation 1 (see Lemma 3), $(N_G[C_1], \ldots, N_G[C_k])$ is a clique path of H', in particular H' is an interval graph. Consequently H' contains some minimal interval completion H of G as required.

Now let H be any minimal interval completion of G such that $N_H(v) = N_G(v)$. We first show that S induces a clique in H. Let D be a component of $G - S$, different from M, such that $N_G(D) = S$. Note that S is a v, u-minimal separator of G, for some $u \in D$. Let T be a minimal v, u separator of H such that $T \subseteq N_H(v)$. Clearly T exists because u and v are non-adjacent in H. We claim that $S \subseteq T$. For each vertex $s \in S$, there is a u, v path of G contained in $D \cup \{v, s\}$. This path intersects $N_G(v)$ only in s, so also in the graph H the only possible intersection between T and the path is s. It follows that $s \in T$, so $S \subseteq T$. The minimal separator T induces a clique in H by Lemma 2. Hence S also induces a clique in H. Note that, by definition of a moplex, $M \cup S$ also induces a clique in H.

For each i, $1 \leq i \leq k$ let $H_i = H[N_G[C_i]]$. Let H'' be the graph with vertex set V and edge set $E(H_1) \cup E(H_2) \cup \cdots \cup E(H_k)$. Therefore $G \subseteq H'' \subseteq H$. We will construct a clique path P of H'', showing that H'' is an interval graph. By minimality of H, this implies that $H'' = H$. Moreover, the clique path P will have $M \cup S = N_G[M]$ as one of its end cliques.

Let $S_i = N_G(C_i)$. By Equation 2, the vertices of S_{i-1} are adjacent to all vertices of $C_i - S_{i-1}$ in the graph G, so also in H_i. Combined with the fact that $S_{i-1} \subseteq S$ induces a clique in H we have that S_{i-1} is contained in each maximal clique of H_i. We claim that for each i, $1 \leq i < k$, there exists a clique path of H_i such that S_i is contained in the rightmost clique of P_i. Indeed, the graph $H_i^+ = H[C_i \cup S \cup M]$ is an interval graph and $M \cup S$ is one of its maximal cliques. Take any clique path P_i^+ of H_i^+, we prove that $M \cup S$ is an end clique. By contradiction, let x (resp y) be a vertex appearing in the clique left (resp. right) to $S \cup M$, but not appearing in $S \cup M$. By the properties of a clique path, $S \cup M$ must separate x and y in H_i^+. This contradicts the fact that $x, y \in C_i$ and there exists an x, y-path in $G[C_i]$. So the only possibility is that $S \cup M$ is at an end of P_i^+. Since $S_i \subseteq S$ and every vertex of S has a neighbour in M, S_i is contained in the clique next to $S \cup M$ in P_i^+. The clique path P_i of H_i obtained by removing $S \cup M$ from P_i^+ has the required property. Eventually, by concatenating the clique paths $P_1, P_2, \ldots, P_k$, it is easy to check that we obtain a clique path P of H''. Indeed if a vertex x appears in the subpaths P_i and P_j with $i < j$, then $x \in N_G[C_i] \cap N_G[C_j] = S_i$ (see Equation 1). By Equation 2, x appears in every clique of P_k, for each $k, i < k \leq j$. Since H_k is the complete graph with vertex set $N_G[M] = S \cup M$, the clique path P has $S \cup M$ as rightmost clique. □

Theorem 5. *Let G be a non-complete graph and v be a LexBFS-terminal moplexian vertex of G. For any minimal interval completion H of G such that $N_G(v) = N_H(v)$, there is an interval ordering of H starting with v.*

Proof. By Lemma 4, there exists a clique path of H such that the left-most clique is $M \cup S$, where $S = N(M)$. We can reverse this path so that $S \cup M$ becomes the leftmost clique of the clique path P. By construction, H has no fill edges incident to v, in particular the v only appears in the left-most clique of P. By Remark 1, there is an interval ordering of H starting with v. □

3.2 A family of nice orderings

Notation 1 *We denote by $\rho = (v_1, \ldots, v_k)$ a prefix, and $R = V \setminus \mathrm{V}(\rho)$. Let* Nxt *be a non-empty subset of R, such that $\mathrm{Nxt} = N_G(v_i) \cap R$ for some $v_i \in \mathrm{V}(\rho)$ and* Nxt *is inclusion-minimal for this property. We denote $R \setminus \mathrm{Nxt}$ by* Rst.

Lemma 5. *Let σ be a refinement of $\rho \bullet \mathrm{Nxt} \bullet \mathrm{Rst}$ and σ' be a refinement of $\rho \bullet R$ such that $G(\sigma') \subseteq G(\sigma)$. Then σ' also is a refinement of $\rho \bullet \mathrm{Nxt} \bullet \mathrm{Rst}$.*

Proof. Let $v_i \in \mathrm{V}(\rho)$ such that $\mathrm{Nxt} = R \cap N_G(v_i)$. Suppose that σ' is not a refinement of $\rho \bullet \mathrm{Nxt} \bullet \mathrm{Rst}$, so there is some vertex $u \in \mathrm{Rst}$ and a vertex $w \in \mathrm{Nxt}$ such that u appears before w in σ'. Since u appears in σ after all vertices of Nxt, v_i and u are not adjacent in $G(\sigma)$. Now in σ', u appears after v_i and before w. Since $\mathrm{Nxt} \subseteq N_G(v_i)$, v_i and w are adjacent in G and therefore v_i and u are adjacent in $G(\sigma')$ – a contradiction. □

Lemma 6. *Consider two vertex orderings σ and σ' of G that are refinements of $\rho \bullet \mathrm{Nxt} \bullet \sigma_{\mathrm{Rst}}$, where σ_{Rst} is an ordering of* Rst. *That is to say, σ and σ' differ only by a permutation of* Nxt. *Let u, v be two vertices adjacent in $G(\sigma')$ but non-adjacent in $G(\sigma)$. Then both $u, v \in \mathrm{Nxt}$.*

Proof. By construction of $G(\sigma)$ and $G(\sigma')$, at least one of the vertices u, v are in Nxt. By contradiction, suppose that the other is not in Nxt.

First we consider the case when $u \in \mathrm{V}(\rho)$ and $v \in \mathrm{Nxt}$. Suppose that u has a neighbour $u' \in \mathrm{Rst}$. In both $G(\sigma)$ and $G(\sigma')$ all vertices of Nxt are adjacent to u as they appear after u and before u' in the corresponding ordering – a contradiction. So $N_G(u) \cap R \subseteq \mathrm{Nxt}$. By definition (minimality) of Nxt, either $\mathrm{Nxt} \subseteq N_G(u)$ or $\mathrm{Nxt} \cap N_G(u) = \emptyset$. Clearly, in the first case Nxt is contained in the neighborhood of u in both $G(\sigma)$ and $G(\sigma')$. In the second, for both $G(\sigma)$ and $G(\sigma')$ the vertex u has no neighbours in Nxt – a contradiction.

It remains to consider the situation when $u \in \mathrm{Nxt}$ and $v \in \mathrm{Rst}$. Since u and v are adjacent in $G(\sigma')$, there is a neighbour v' of u in G, appearing after v in σ'. But u, v, u' appear in the same order in σ, so u and v are adjacent in $G(\sigma)$ – a contradiction. □

3.3 Nice orderings : a sufficient condition

Notation 2 *Let σ be a vertex ordering of G and let ρ be a prefix of σ. We denote by T the set of vertices of* Nxt *having neighbours in* Rst. *G_{Nxt} denotes the graph obtained from $G[\mathrm{Nxt}]$ by adding a dummy vertex d_1, adjacent to each vertex of T, and a dummy vertex d_2 adjacent only to d_1. The graph G^+_{Nxt} is obtained from G_{Nxt} by completing T into a clique. Given a clique path P of H, let $P[\mathrm{Nxt}]$ denote the clique path of $H[\mathrm{Nxt}]$ obtained by restricting all the bags of P to their intersections with* Nxt *and then removing the redundant ones (leaving only unique maximal cliques of $H[\mathrm{Nxt}]$).*

Theorem 6. *Let σ be a vertex ordering of G with the following properties:*

1. *σ starts with a LexBFS-terminal vertex v_1.*
2. *For any non-empty prefix $\rho = (v_1, \ldots, v_i)$*

– *σ respects ρ, i.e. σ is a refinement of $\rho \bullet \mathrm{Nxt} \bullet \mathrm{Rst}$,*
– *the next vertex in σ is a LexBFS-terminal vertex of G^+_{Nxt} obtained by running LexBFS starting from d_2.*

Then σ is a nice ordering.

Proof. Suppose that $\sigma = (v_1, \ldots, v_n)$ is not nice and let σ' be an ordering such that $H' = G(\sigma')$ is a minimal interval completion of G strictly contained in $H = G(\sigma)$. Take σ' such that the maximal common prefix ρ of σ and σ' is the longest possible.

Claim. ρ is not empty.

The first vertex v_1 of σ is LexBFS-terminal. $N_G(v_1) = N_{G(\sigma)}(v_1)$, since σ respects the prefix (v_1) and thus the neighbours of v_1 in G appear right after v_1 in σ. So the Claim follows by Theorem 5.

Let v (resp. u) be the vertex right after ρ in σ (resp. in σ'). By Lemma 5, we have:

Claim (1). σ' is a refinement of $\rho \bullet \mathrm{Nxt} \bullet \mathrm{Rst}$, in particular $u \in \mathrm{Nxt}$.

Claim (2). Let σ'' be any refinement of $\rho \bullet \mathrm{Nxt} \bullet \mathrm{Rst}$ and $H'' = G(\sigma'')$. Let $P^{''} = P(G, \sigma'')$ (see Remark 1). Then $H''[\mathrm{Nxt}]$ is an interval completion of $G[\mathrm{Nxt}]$, where the clique path $P''[\mathrm{Nxt}]$ has the set $T = N_G(\mathrm{Rst}) \cap \mathrm{Nxt}$ contained in one of the end-cliques. In particular, T is a clique in H''.

Clearly, $P^{''}[\mathrm{Nxt}]$ is a clique path of $H^{''}[\mathrm{Nxt}]$. The last clique contains T, since the corresponding intervals in the model intersect the interval of a vertex in Rst.

Claim (3). $H'[\mathrm{Nxt}]$ is an interval completion of $G[\mathrm{Nxt}]$, minimal with respect to the property expressed in the previous claim.

Since σ' defines a minimal interval completion H' of G, σ' has to yield $H'[\mathrm{Nxt}]$ minimal with this property. Suppose it is not minimal, and let $H^{'''}[\mathrm{Nxt}]$ be the corresponding completion strictly included in $H^{'}[\mathrm{Nxt}]$. Then we can take the corresponding clique path $P^{'''}[\mathrm{Nxt}]$ to create an interval order $\sigma^{'''}_{\mathrm{Nxt}}$ of $H^{'''}[\mathrm{Nxt}]$ (see Remark 1). By Lemma 6, $\sigma^{'''} = \rho \bullet \sigma^{'''}_{\mathrm{Nxt}} \bullet \sigma^{'}_{\mathrm{Rst}}$ yields $H^{'''} = G(\sigma^{'''})$ strictly contained in $H^{'}$. A contradiction with minimality of $H^{'}$.

Following Notation 2, let H'_{Nxt} be obtained from $H'[\mathrm{Nxt}]$ by adding a dummy vertex d_1 adjacent to the vertices of T and a vertex d_2 adjacent to d_1.

Claim (4). H'_{Nxt} is a minimal interval completion of G^+_{Nxt}.

Let P'_{Nxt} denote the clique path of H'_{Nxt}, obtained from $P'[\mathrm{Nxt}]$ by ading two bags $q_1 = T \cup \{d_1\}$ and $q_2 = \{d_1, d_2\}$ after the clique containing T (see Claim 2). It is a clique path indeed, so H'_{Nxt} is an interval completion of G^+_{Nxt}. Suppose it is not minimal. So there is a minimal one $H^{''}_{\mathrm{Nxt}}$ strictly included in H'_{Nxt}. Notice that this graph has a clique path $P^{''}_{\mathrm{Nxt}}$, that also has $--q_1--q_2$ at an end. Indeed, the moplex $M = d_2$ satisfies the conditions of Lemma 4 in $H^{''}_{\mathrm{Nxt}}$, so there is a clique path of $H^{''}_{\mathrm{Nxt}}$ with $\{d_1, d_2\}$ as one of the end-cliques. Therefore $P^{''}[\mathrm{Nxt}]$, obtained by removing $--q_1--q_2$ from $P^{''}_{\mathrm{Nxt}}$, is a clique path of $H^{''}[\mathrm{Nxt}]$ with T contained in one of the end-cliques. Which contradicts Claim 3, since $H^{''}[\mathrm{Nxt}]$ is a strict subgraph of $H'[\mathrm{Nxt}]$.

Claim (5). There is an interval ordering of H'_{Nxt} starting with v.

Let us prove that $N_{H'_{\mathrm{Nxt}}}(v) = N_{G^+_{\mathrm{Nxt}}}(v)$. Indeed if $v \in T$, since v is the last vertex encountered by LexBFS launched on G^+_{Nxt} from d_2, we have $T = \mathrm{Nxt}$. In this case the neighborhood of v in both graphs is $T \setminus \{v\} \cup \{d_1\}$, and the equality follows.

Now if $v \notin T$ then $N_G(v) \cap R \subset \mathrm{Nxt}$. By second condition of the theorem, σ respects $\rho \bullet v$, so $N_G(v) \cap \mathrm{Nxt}$ is put before $R \setminus N_G(v)$ in σ and $N_{H[\mathrm{Nxt}]}(v) = N_{G[\mathrm{Nxt}]}(v)$. Therefore $N_{H'_{\mathrm{Nxt}}}(v) = N_{G^+_{\mathrm{Nxt}}}(v)$, since

$$N_{G^+_{\mathrm{Nxt}}}(v) \subseteq N_{H'_{\mathrm{Nxt}}}(v) \subseteq N_{H_{\mathrm{Nxt}}}(v) = N_{G_{\mathrm{Nxt}}}(v) \subseteq N_{G^+_{\mathrm{Nxt}}}(v).$$

The claim follows from Theorem 5 and Claim 4.

Claim (6). There is an ordering σ'', with $G(\sigma'') = G(\sigma')$, sharing a longer prefix with σ – a contradiction.

We restrict the ordering from the previous claim to Nxt and obtain σ''_{Nxt}. Let $\sigma'' = \rho \bullet \sigma''_{\mathrm{Nxt}} \bullet \sigma'_{\mathrm{Rst}}$. By Lemma 6, $G(\sigma'') = G(\sigma')$. So σ'' defines the same completion and shares a longer prefix. Which contradicts the choice of σ'.

This achieves the proof of our theorem. □

4 The algorithm

Theorem 7. *There is an $\mathcal{O}(nm)$-time algorithm computing a minimal interval completion of an arbitrary graph.*

Proof. We prove that the algorithm `MIC_Ordering` of Figure 2 computes in $\mathcal{O}(nm)$ time a vertex ordering satisfying the conditions of Theorem 6.

Clearly the first vertex v_1 is a LexBFS-terminal vertex, implying the first condition of Theorem 6. The initialization of the ordered partition ensures that all neighbours of v_1 in G appear contiguously and right after v_1 in σ. Therefore σ is a refinement of $v_1 \bullet N_G(v_1) \bullet V \setminus N_G[v_1]$.

The algorithm maintains an ordered partition of the vertex set of G. At each step i the set Nxt corresponds like in Notation 1 to the prefix $\rho = (v_1, \ldots, v_i)$ as required. By contradiction suppose there exists $j < i$ such that $N(v_j)$ is strictly contained in Nxt. Then at step j, the class C of the ordered partition containing Nxt has been split in $C \cap N_G(v_j)$ and $C \setminus N_G(v_j)$ – contradicting the fact that Nxt is a class of the ordered partition at the step i.

Unlike the Theorem 6, our algorithm computes the vertex v_i by launching LexBFX from d_2 on the graph G_{Nxt} and not G^+_{Nxt}. The reason is related to the running time. Indeed G_{Nxt} has $\mathcal{O}(n+m)$ edges, while if we compute G^+_{Nxt}, the number of edges of G^+_{Nxt} might be up to $\Omega(n^2)$. Nevertheless we prove that v_i is also a LexBFS-terminal vertex obtained by using G^+_{Nxt} instead of G_{Nxt}. Let n' be the number of vertices of G_{Nxt}. When the vertex d_1 is numbered (with number $n'-1$) by LexBFS on G_{Nxt}, all vertices of T are labeled $(n-1)$. Then all vertices of T are numbered before the vertices of $\mathrm{Nxt} \setminus T$. The same would have happened by running LexBFS on G^+_{Nxt}. Moreover, in G^+_{Nxt} any numbering of T is valid in this case. So the LexBFS numbering on G_{Nxt} is also a LexBFS numbering on G^+_{Nxt}.

Together with the way that algorithm maintains an ordered partition, it implies the second condition of Theorem 6.

Each iteration of the **for** loop must be performed in $\mathcal{O}(m)$ time. The update of the order partition (the last three lines of the loop) can be easily done in $\mathcal{O}(n)$ time. (We point out that, using more envolved techniques for partition refinement [7, 8], this step could even be done in $O(|N_G(v_i)|$ time.) The choice of the vertex v_i is made by running LexBFS on the graph G_{Nxt}. Since this graph is of size $\mathcal{O}(n+m)$. The $\mathcal{O}(nm)$-time for computing σ follows.

The function **IntervalModel** constructs an interval model of $G(\sigma)$ based on the ordering σ like in Remark 1. A single pass along $\sigma = (v_1, \ldots, v_n)$ is enough to assign to every vertex v_i the interval $[i : j]$, where j is the biggest index such that $v_i v_j \in E(G)$. This can be done in $\mathcal{O}(deg_G(v))$-time per vertex v, where $deg_G(v)$ is the degree of v in G. In total, it gives $\mathcal{O}(n+m)$-time for computing the interval model. □

Algorithm `MIC_Ordering`
Input: $G = (V, E)$ connected
Output: a nice ordering σ and the corresponding interval model
let v_1 be the last vertex encountered by `LexBFS(`G`)`
$\rho := (v_1)$
$\mathrm{Nxt} = N_G(v_1); \mathrm{Rst} := V \setminus N_G[v_1]$
$OP := v_1 \bullet \mathrm{Nxt} \bullet \mathrm{Rst}$
for $i := 2$ **to** n **do**
    let Nxt be the class appearing after ρ in OP
    let v_i be the last vertex encouneterd by `LexBFS`
        launched on G_{Nxt} starting from d_2 (see Theorem 6)
    $\rho = \rho \bullet v_i$
    if $|\,\mathrm{Nxt}\,| \geq 2$ **then**
        replace Nxt in OP by $v_i \bullet (\mathrm{Nxt} \setminus \{v_i\})$
    let C be the last class of OP such that $N_G(v_i) \cap C \neq \emptyset$
    if $C \setminus N_G(v_i) \neq \emptyset$ **then**
        replace C in OP by $(C \cap N_G(v_i)) \bullet (C \setminus N_G(v_i))$
$\sigma := OP$
IntervalModel(σ)

Fig. 2. Algorithm `MIC_Ordering`

5 Conclusion

We give in this paper an $\mathcal{O}(nm)$ time algorithm computing a minimal interval completion of an arbitrary input graph. The algorithm is based on the notion of nice orderings, which characterize a minimal interval completion, and on Theorem 6 which gives a sufficient condition for a nice ordering. We point out that there are nice orderings satisfying the conditions of Theorem 6, which cannot be produced by the algorithm. Such examples can be easily obtained when the input graph is a cycle. In particular an ordering produced by our algorithm is always a breadth-first search ordering, which is not required by the theorem.

There are two very natural directions for further research. One is to obtain a faster algorithm for the minimal interval completion problem. In our algorithm, each time when we choose a new vertex, we need LexBFS as the tie-break rule. A faster choice would improve the running time of the algorithm: just maintaining the ordered partition can be done in linear time [7, 8]. A naive technique would consist of doing only one sweep of LexBFS and then choosing, at each step, the vertex of Nxt with minimum LexBFS number. Unfortunately this approach does not produce a minimal interval completion.

The second important question is to characterize all nice orderings. For the minimal triangulation problem, the perfect elimination orderings (which play the same role as the nice orderings here) have been completely characterized. In our case, we have examples of nice orderings that do not satisfy the conditions of Theorem 6.

References

1. A. Berry, J. P. Bordat, *Separability Generalizes Dirac's Theorem.* Discrete Applied Mathematics, 84(1-3): 43-53, 1998.
2. A. Berry, J. P. Bordat, *Local LexBFS Properties in an Arbitrary Graph.* Proceedings of Journes Informatiques Messines, 2000. http://www.isima.fr/berry/lexbfs.ps.
3. H. L. Bodlaender, *A Linear-Time Algorithm for Finding Tree-Decompositions of Small Treewidth.* SIAM Journal on Computing, 25(6):1305-1317, 1996.
4. L. Cai, *Fixed-Parameter Tractability of Graph Modification Problems for Hereditary Properties.* Information Processing Letters, 58(4):171-176, 1996.
5. P. C. Gilmore and A. J. Hoffman, *A characterization of comparability graphs and of interval graphs.* Canadian Journal of Mathematics, 16:539-548, 1964.
6. M. C. Golumbic, *Algorithmic Graph Theory and Perfect Graphs.* Academic Press, 1980.
7. M. Habib, C. Paul, L. Viennot, *Partition Refinement Techniques: An Interesting Algorithmic Tool Kit.* International Journal of Foundations of Computer Science, 10(2): 147-170, 1999.
8. M. Habib, R. M. McConnell, C. Paul, L. Viennot, *Lex-BFS and partition refinement, with applications to transitive orientation, interval graph recognition and ones testing.* Theoretical Computer Science, 234(1-2): 59-84, 2000.
9. P. Heggernes, F. Mancini, *Minimal Split Completions of Graphs.* Proceedings of LATIN 2006, Lecture Notes in Computer Science, 3887:592-604, 2006.
10. P. Heggernes, F. Mancini, C. Papadopoulos *Minimal Comparability Completions.* Tech. Report, University of Bergen, 2006, http://www.ii.uib.no/publikasjoner/texrap/pdf/2006-317.pdf
11. P. Heggernes, K. Suchan, I. Todinca,Y. Villanger, *Minimal Interval Completions.* Proceedings of the 13th Annual European Symposium on Algorithms - ESA 2005, Lecture Notes in Computer Science, 3669:403-414, 2005.
12. P. Heggernes, J. A. Telle, Y. Villanger, *Computing minimal triangulations in time $O(n^{\alpha} logn) = o(n^{2.376})$.* Proceedings of the 16th Annual ACM-SIAM Symposium on Discrete Algorithms - SODA 2005, SIAM, 907-916, 2005.
13. D. Kratsch, J. Spinrad, *Minimal fill in $\mathcal{O}(n^{2.69})$ time.* To appear in Discrete Applied Mathematics.
14. H. Kaplan, R. Shamir, *Pathwidth, Bandwidth, and Completion Problems to Proper Interval Graphs with Small Cliques.* SIAM Journal on Computing, 25(3): 540-561, 1996.
15. H. Kaplan, R. Shamir, R. E. Tarjan, *Tractability of Parameterized Completion Problems on Chordal, Strongly Chordal, and Proper Interval Graphs.* SIAM Journal on Computing, 28(5): 1906-1922, 1999.
16. B. Monien, *The bandwidth minimization problem for caterpillars with hair length 3 in NP-complete.* SIAM Journal on Algebraic and Discrete Methods, 7:505-512, 1986.
17. S. Olariu, *An optimal greedy heuristic to color interval graphs.* Information Processing Letters, 37(1): 21–25, 1991.
18. I. Rappaport, K. Suchan, I. Todinca, *Minimal proper interval completions.* To appear in Proceedings of the 32nd Workshop on Graph-Theoretic Concepts in Computer Science (WG'06), 2006. http://www.univ-orleans.fr/SCIENCES/LIFO/prodsci/rapports/RR2006.htm.en.
19. D. Rose, R.E. Tarjan, and G. Lueker, *Algorithmic aspects of vertex elimination on graphs.* SIAM J. Comput., 5:146–160, 1976.