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 proper interval completions

Ivan RAPAPORT
Universidad de Chile, Santiago, Chile
Karol SUCHAN
LIFO, Université d'Orléans
Ioan TODINCA
LIFO, Université d'Orléans

Rapport N^o **2006-02**

Minimal proper interval completions*

Ivan Rapaport[1], Karol Suchan[23], and Ioan Todinca[2]

[1] Departamento de Ingeniería Matemática and Centro de Modelamiento Matemático, Universidad de Chile, Santiago, Chile,
irapapor@dim.uchile.cl
[2] LIFO, Université d'Orléans, 45067 Orléans Cedex 2, France,
Karol.Suchan,Ioan.Todinca@univ-orleans.fr
[3] Department of Discrete Mathematics, Faculty of Applied Mathematics, AGH - University of Science and Technology, Cracow, Poland

March 8, 2006

Abstract. Given an arbitrary graph $G = (V, E)$ and a proper interval graph $H = (V, F)$ with $E \subseteq F$ we say that H is a *proper interval completion* of G. The graph H is called a *minimal proper interval completion* of G if, for any sandwich graph $H' = (V, F')$ with $E \subseteq F' \subset F$, H' is not a proper interval graph. In this paper we give a $\mathcal{O}(n + m)$ time algorithm computing a minimal proper interval completion of an arbitrary graph. The output is a proper 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 cliquesize of H). If instead of minimizing the cliquesize of 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 (resp. a proper interval) graph, we say that H is an interval completion (resp. proper interval completion) of G. Recall that an interval graph is a *proper interval graph* if it has an interval model such that no interval is properly contained into another. The *pathwidth* of G can be defined as $\min(\omega(H)) - 1$, where the minimum is taken over all interval completions of G. The minimum number of edges that we need to add for obtaining an interval completion is called the *profile* of the graph.

Proper interval graph completions have been discussed in [12]. Independently, Kaplan et al. [12] and Cai [3] show that the problem of computing the minimum number of edges $|E'|$ such that $H = (V, E \cup E')$ becomes a proper interval graph is fixed parameter tractable. The problem is addressed as the "proper interval graph completion problem", motivated by applications to genetics. The *bandwidth* of a graph is usually expressed as follows. Consider an ordering (also called layout) $\sigma = (v_1, \ldots, v_n)$ of the vertices of G. The width of the layout is $\max\{|i-j| \mid v_i, v_j \text{ adjacent in } G\}$. The bandwidth of G is the minimum width over all layouts of G. It has been proved in [11] that the bandwidth of G is also equal to $\min(\omega(H)) - 1$, the minimum being taken over all proper interval completions of G (see also Section 2 for the relationship between layouts and proper interval completions). The bandwidth problem

* Partially supported by Programs Conicyt "Anillo en Redes" (I.R.) and Ecos-Conicyt (I.R., I.T).

for graphs, motivated by the bandwidth minimization problem for matrices, is one of the few graph problems NP-hard even for the class of trees [13]. Computing the bandwidth is also $W[t]$-hard for all t, thus unlikely to be fixed parameter tractable.

For each of the parameters cited above, 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*, *minimal proper interval completion*) if no proper subgraph of H is a triangulation (interval completion, proper 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, similarly, the study of other types of minimal completions might bring new powerful 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 ([10], running in $\mathcal{O}(n^{2.69})$ time) and Heggernes, Telle and Villanger ([9], 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 later algorithm is the fastest up to now for the minimal triangulation problem.

A first polynomial algorithm solving the minimal interval completion problem has been given in [8]. Heggernes and Mancini [7] gave a linear time algorithm for computing a minimal embedding into split graphs.

Our result. We study the minimal proper interval completion problem. Our main result is a linear time algorithm computing a minimal proper interval completion of an arbitrary graph. One of the main tools is a special ordering of the proper interval graph, called bicompatible ordering [14]. Its role is similar to the simplicial elimination schemes for chordal graph. We define a family of orderings such that the associated proper interval graph is a minimal proper interval completion. Eventually, we give a linear-time algorithm (based on a BFS) computing such an ordering. The ordering can be efficiently transformed into a proper 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 subgraph of $G' = (V', E')$ (i.e. $V \subseteq V'$ and $E \subseteq E'$) we write $G \subseteq G'$. The *neighborhood* of a vertex v in G is $N_G(v) = \{u \mid \{u, v\} \in E\}$. Similarly, for a set $A \subseteq V$, $N_G(A) = \bigcup_{v \in A} N_G(v) \setminus A$. As usual, the subscript is sometimes omitted. For a set of vertices $A \subset V$, $G[A]$ is the *subgraph* of G *induced* by A, that is the graph (A, E_A), where $E_A = \{\{x, y\} \mid \{x, y\} \subseteq A \text{ and } \{x, y\} \in E\}$. A vertex v is called *simplicial* if $N(v)$ is a clique of G. A *perfect elimination ordering* of V is an ordering $\sigma = (v_1, \ldots, v_n)$ of V with the property that v_i is a simplicial vertex of $G[\{v_i, v_{i+1}, \ldots, v_n\}]$, for every i, $1 \leq i \leq n$.

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. A graph G is interval if and only is has a *clique path* CP, i.e. a path whose vertex set is the set of all maximal cliques of G, such that for each vertex v of G, the subgraph of CP induced by the maximal cliques containing v is connected. Taking this induced interval for each vertex of G yields an interval model. If an interval graph G has an interval model where no interval is properly contained in another, then the graph is called a *proper interval* graph.

Proper interval graphs can also be characterized as unit interval graphs (all intervals have equal length) or claw-free interval graphs (interval graphs without induced $K_{1,3}$). See e.g. [4] for more details. For our purpose, we use their characterization in terms of *bicompatible orderings.*

Definition 1 ([14]). *Let $G = (V, E)$ be a graph and $\sigma = (v_1, v_2, \ldots, v_n)$ be an ordering of its vertices. If both σ and the reverse of σ is a perfect elimination ordering, then σ is called* bicompatible*.*

Theorem 1 ([14]). *H is a proper interval graph if and only if there exists a bicompatible ordering of its vertices.*

The following statement can be considered as an equivalent definition for bicompatible orderings. In our work we rather use this characterization.

Lemma 1 (Characterization of bicompatible orderings [14]). *Let $H = (V, F)$ be a proper interval graph. Then $\sigma = (v_1, v_2 \ldots, v_n)$ is a bicompatible ordering of H if and only if $\{v_i, v_l\} \in F$ implies that $\{v_j, v_k\} \in F$ for all i, j, k, l, $1 \leq i \leq j < k \leq l \leq n$.*

Definition 2. *A tuple of disjoint subsets of V, $P = (P_1, \ldots, P_k)$ whose union is exactly V is called an* ordered partition *of V. A* refinement *of P is an ordered partition P' obtained by replacing each set P_i by an ordered partition of P_i. We write $P' \preccurlyeq P$.*

Definition 3. *Given an ordered partition $P = (P_1, \ldots, P_k)$, any tuple $P' = (P_1, \ldots, P_j)$, with $0 \leq j \leq k$, is called a* prefix *of P. We use $\mathrm{V}(P')$ to denote $\bigcup\{P_i \mid 1 \leq i \leq j\}$.*

In the particular case where $P = (P_1)$, we simply write P_1. Moreover if P_1 is formed by a single vertex x, we write x instead of $\{x\}$. Given two tuples $P' = (P_1, \ldots, P_k)$, $P'' = (P_{k+1}, \ldots, P_{k+l})$ we write $P' \bullet P''$ to denote their concatenation $P = (P_1, \ldots, P_k, P_{k+1}, \ldots, P_{k+l})$.

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

Definition 4. *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*

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

Lemma 2. *$G(\sigma)$ is a proper interval graph.*

Proof. It is a direct consequence of Lemma 1 and Theorem 1.

Remark 1. Let $\sigma = (v_1, v_2 \ldots, v_n)$ be a bicompatible ordering of a proper interval graph $G = (V, E)$. Let $(v_{l_1}, \ldots, v_{l_j})$, where l_c is monotonically increasing, be the list of vertices v_l such that $N(v_l) \setminus N(\{v_1, \ldots, v_{l-1}\}) \neq \emptyset$. Let v_{r_c} be the last neighbor of v_{l_c} in σ, for $1 \leq c \leq j$. For each $c, 1 \leq c \leq j$ let $K_c = [v_{l_c} : v_{r_c}]$ be the set of vertices appearing between v_{l_c} and v_{r_c} in σ. The tuple $(K_1, \ldots, K_j)$ forms a clique path of $G(\sigma)$.

Theorem 2. *Let $G = (V, E)$ be an arbitrary graph and $H = (V, F)$ be a minimal proper interval completion of G. Then there is an ordering σ such that $H = G(\sigma)$.*

Proof. By Theorem 1, there is an ordering σ of V bicompatible for H. As a straight consequence of Definition 4 and Lemma 1, $E(G(\sigma)) \subseteq E(H)$. By Lemma 2, $G(\sigma)$ is also a proper interval graph. Thus, by minimality of H, we deduce that $E(G(\sigma)) = E(H)$. □

Definition 5. *An ordering σ is called* nice *if $G(\sigma)$ is a minimal proper interval completion of G. Any prefix of a nice ordering is also called* nice.

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. 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 $N(C) = N(D) = S$.

Definition 6 ([1]). *A* moplex *is a maximal clique module M such that $N(M)$ is a minimal separator of G. A vertex $v \in M$ of G is called* moplexian.

Proposition 1. *Let M be a moplex of G and $v \in M$. There exist a nice ordering σ starting with v such that the neighborhood of v in $G(\sigma)$ is exactly the neighborhood of v in G. Moreover, for any minimal interval completion H' of G such that $N_G(v) = N_{H'}(v)$, there exists an ordering σ', starting with v and such that $H' = G(\sigma')$.*

Proof. Let M be a moplex such that $v \in M$ (actually this moplex is unique) and let H be the graph obtained from G by completing $V \setminus M$ into a clique. We first show that H is a proper interval graph. Let $S = N(M)$. By definition of a moplex and by construction of H, the graph H is formed by two cliques, namely $M \cup S$ and $V \setminus M$. Their intersection is exactly S. Clearly H is an interval graph. Moreover it has no independent set of size greater that 2, in particular it has no induced $K_{1,3}$. Hence H is interval and claw-free, so H is a proper interval graph (see [4]). In particular there is a minimal proper interval completion of G contained in H.

Consider any minimal proper interval completion H' of G such that $N_G(v) = N_{H'}(v)$ (H' exists by the previous remark). By Theorem 1, there exists an ordering σ' such that $H' = G(\sigma')$. If all vertices appearing before v in σ' are elements of the module M, we can permute v and the first element of σ' without changing the graph $G(\sigma')$. Similarly, if all vertices appearing after v are in M, we reverse σ' and then permute v and the first vertex. In both cases v becomes the first vertex of σ'.

It remains to consider the case when there are two vertices $a, b \notin M$, such that $a < v < b$ in σ'. There is a path from a to b in G, such that all vertices of the path are in $V \setminus M$. Consequently there are two consecutive vertices of the path, say a' and b', such that $a' < v < b'$ in the ordering σ'. Thus $\{v, a'\}$ and $\{v, b'\}$ are edges of H'. Since $a', b' \notin M$, by construction of H' we must have $a', b' \in S$. Recall that S is a minimal separator, thus there are two connected components C and D of $G - S$ such that $N(C) = N(D) = S$. At least one of them, say C, is different from M. Let μ be a path from a' to b' in $G[C \cup \{a', b'\}]$, not using the edge $\{a', b'\}$. Like above, there are two consecutive vertices a'' and b'' of μ with $a'' < v < b''$ in σ. Hence v

is adjacent in H' to both a'' and b''. At least one of a'', b'' is in C, contradicting the fact that H' has no edges between v and $V \setminus (M \cup S)$. $\square$

A moplexian vertex always exists an can be found efficiently.

Theorem 3 ([1]). *Every graph has a moplexian vertex. Such a vertex can be found in $O(n+m)$ time. More precisely, the algorithm LexBFS ends on a moplexian vertex.*

3.2 A family of nice orderings

Definition 7. *Let ρ be a non empty prefix. We denote by* $\mathrm{First}(\rho)$ *the first vertex in ρ having a neighbor in $V \setminus \mathrm{V}(\rho)$. We define the strong neighborhood (denoted $\mathrm{N_S}(\rho)$), weak neighborhood ($\mathrm{N_W}(\rho)$) and non-neighborhood $\overline{\mathrm{N}}(\rho)$ as follows:*

- $\mathrm{N_S}(\rho) = N(\mathrm{First}(\rho)) \setminus \mathrm{V}(\rho)$,
- $\mathrm{N_W}(\rho) = N(\mathrm{V}(\rho)) \setminus \mathrm{N_S}(\rho)$,
- $\overline{\mathrm{N}}(\rho) = V \setminus (\mathrm{V}(\rho) \cup \mathrm{N_S}(\rho) \cup \mathrm{N_W}(\rho))$.

Definition 8. *We say that an ordering σ* respects *a prefix ρ if σ is a refinement of $\rho \bullet (\mathrm{N_S}(\rho), \mathrm{N_W}(\rho), \overline{\mathrm{N}}(\rho))$.*

Our goal is to show that if ρ is a nice prefix starting with a moplexian vertex, then there is a nice ordering respecting it. This is a first step towards the extension of a nice prefix by adding a new vertex. Also note that a BFS ordering respects all its prefixes. Actually our construction of a nice ordering will be based on a BFS starting from a moplexian vertex.

Lemma 3. *Let σ and σ' be two orderings with a common prefix ρ and such that $G(\sigma') \subseteq G(\sigma)$. Suppose that σ is a refinement of $\rho \bullet (\mathrm{N_S}(\rho), \mathrm{N_W}(\rho) \cup \overline{\mathrm{N}}(\rho))$. Then σ' is also a refinement of $\rho \bullet (\mathrm{N_S}(\rho), \mathrm{N_W}(\rho) \cup \overline{\mathrm{N}}(\rho))$.*

Proof. Assume that both sets $\mathrm{N_S}(\rho)$ and $\mathrm{N_W}(\rho) \cup \overline{\mathrm{N}}(\rho)$ are not empty, otherwise the conclusion is true for any σ' starting with ρ. Let v_f denote $\mathrm{First}(\rho)$.

By contradiction suppose that there are two vertices $a \in \mathrm{N_S}(\rho)$ and $b \in \mathrm{N_W}(\rho) \cup \overline{\mathrm{N}}(\rho)$ such that $v_f < b < a$ in the ordering σ'. Therefore $\{v_f, b\}$ is an edge of $G(\sigma')$. If $G(\sigma)$ contained the edge $\{v_f, b\}$, then there are two adjacent vertices v' and a' of G such that $v' \le v_f < b \le a'$ in the ordering σ. By definition of $v_f = \mathrm{First}(\rho)$ we must have $v' = v_f$. Therefore $a' \in \mathrm{N_S}(\rho)$, contradicting the fact that $\mathrm{N_S}(\rho)$ appears before b in σ.

We conclude that the edge $\{v_f, b\}$ appears in $G(\sigma')$ but not in $G(\sigma)$. $\square$

Lemma 4. *Let σ and σ' be two orderings with a common prefix ρ and such that $G(\sigma') \subseteq G(\sigma)$. Assume that σ respects ρ and let $u \in \mathrm{N_W}(\rho)$, $w \in \overline{\mathrm{N}}(\rho)$. Then u appears before w in σ'.*

Proof. By contradiction, suppose that w appears before u in σ'. Let $u' \in \mathrm{V}(\rho)$ be a neighbor of u. The edge $\{w, u'\}$ is present in $G(\sigma')$, since w is between u' and u in σ'. On the other hand, σ respects ρ, so w appears after $\rho \bullet (\mathrm{N_S}(\rho), \mathrm{N_W}(\rho))$. No element of ρ is adjacent in G to a vertex appearing after w in σ. By construction of $G(\sigma)$, this graph does not contain the edge $\{w, u'\}$.

Lemmas 3 and 4 directly imply the following:

Proposition 2. *Let σ and σ' be two orderings with a common prefix ρ and such that $G(\sigma') \subseteq G(\sigma)$. If σ respects ρ, then σ' also respects ρ.*

Lemma 5. *Let ρ be a non-empty prefix. Let $u, w \in \mathrm{N_S}(\rho)$. Let σ be an ordering that respects $\rho \bullet u$. Let σ' be an ordering, with ρ as a prefix, in which w appears before u. If there is $w' \in (N(w) \cap \overline{\mathrm{N}}(\rho)) \setminus (N(u) \cap \overline{\mathrm{N}}(\rho))$, then the graph $G(\sigma')$ contains an edge not appearing in $G(\sigma)$.*

Proof. If w' is between $\mathrm{First}(\rho)$ and u in σ', then by Definition 4 $\{u, w'\}$ is present in $G(\sigma')$. Else, u is between w and w' in σ' and the same holds. On the other hand, σ respects $\rho \bullet u$, so w' appears after $\rho \bullet (u, \mathrm{N_S}(\rho \bullet u), \mathrm{N_W}(\rho \bullet u))$. No element of $\rho \bullet u$ is adjacent in G to a vertex appearing after w' in σ. By Definition 4, $\{u, w'\}$ is not an edge of $G(\sigma)$.

Lemma 6. *Let $\sigma = (v_1, \ldots, v_n)$ be an ordering of V. Let σ' be obtained from σ by permuting vertices strictly between v_i,v_k in σ. Then every edge in the symmetric difference $E(G(\sigma')) \overline{\cup} E(G(\sigma))$ is incident to a vertex v_j between v_i and v_k in σ.*

Proof. The proof is a straightforward consequence of the construction of $G(\sigma)$ and $G(\sigma')$. □

3.3 Nice orderings : a sufficient condition

Our main combinatorial result is that nice orderings can be obtained from a BFS ordering starting with a moplexian vertex, with an additional tie-break rule.

Theorem 4. *Let $G = (V, E)$ be a graph. Let $\sigma = (v_1, \ldots, v_n)$ be an ordering of V such that v_1 is a moplexian vertex and for each $1 < i < n$:*

1. *σ respects ρ, where $\rho = (v_1, \ldots, v_{i-1})$,*
2. *v_i is such that $N(v_i) \cap \overline{\mathrm{N}}(\rho)$ is inclusion-minimal over all vertices in $\mathrm{N_S}(\rho)$.*

Then σ is a nice ordering.

Proof. Suppose that σ is not a nice ordering and let σ' be an ordering such that $G(\sigma')$ is a strict subgraph of $G(\sigma)$. Take σ' in order to maximize the common prefix of σ and σ'. Let $\rho = (v_1, \ldots, v_p)$ be this maximum common prefix. By construction of σ, all the edges of $G(\sigma)$ incident to v_1 are also edges of G. By Proposition 1, σ' starts with v. Consequently ρ has at least one vertex.

Let $u = v_{p+1}$ be the vertex of index $p+1$ in σ and w be the vertex of index $p+1$ in σ'.

By Proposition 2, σ' respects ρ.

Let σ'' be the ordering obtained from σ' by exchanging u and w. We claim that $G(\sigma'') = G(\sigma')$. By Lemma 6, any edge that might differ from $G(\sigma'')$ to $G(\sigma')$ is adjacent to a vertex between u and w, let I denote this interval. Since σ and σ' respect ρ, we have that $u, w \in \mathrm{N_S}(\rho)$, hence $I \subseteq \mathrm{N_S}(\rho)$. As a consequence of Lemma 5 and by the condition 2 of the theorem, $N(x) \cap \overline{\mathrm{N}}(\rho) = N(u) \cap \overline{\mathrm{N}}(\rho)$ for every $x \in I$. Let z be the last vertex of σ' contained in $N(u) \cap \overline{\mathrm{N}}(\rho)$, if such a vertex exists. In particular z is also the last vertex of σ'' in $N(u) \cap \overline{\mathrm{N}}(\rho)$.

Consider any $y \in \mathrm{V}(I)$. Both in $G(\sigma')$ and $G(\sigma'')$, y is adjacent to all vertices of $\mathrm{V}(\rho)$ appearing after $\mathrm{First}(\rho)$ and has no neighbor appearing strictly before $\mathrm{First}(\rho)$. Since $y \in \mathrm{N_S}(\rho)$, the vertices of $\overline{\mathrm{N}}(\rho)$ adjacent to y in $G(\sigma')$ are precisely the ones appearing before z – or this neighborhood is empty if z does not exist. The same holds for $G(\sigma'')$. Eventually, $\mathrm{N_S}(\rho) \cup \mathrm{N_W}(\rho)$ induces a clique both in $G(\sigma')$ and $G(\sigma'')$. Indeed the last vertex b of $\mathrm{N_S}(\rho) \cup$

$\mathrm{N_W}(\rho)$ in σ' (resp. σ'') is adjacent in G to some vertex a of ρ. Since σ' and σ'' respect ρ, all the vertices of $\mathrm{N_S}(\rho) \cup \mathrm{N_W}(\rho)$ are in between a and b, so they form a clique. That proves that $G(\sigma') = G(\sigma'')$.

We have proved that σ'' and σ have $\rho \bullet u$ as common prefix, and $G(\sigma'') \subseteq G(\sigma)$. This contradicts the choice of σ'. □

4 The algorithm

The algorithm is based on a BFS, see Figure 1. It creates an ordering σ of the vertices like in Theorem 4 and then returns a proper minimal model of the minimal proper interval completion $G(\sigma)$.

Theorem 5. *There is a linear time algorithm that, given an arbitrary graph G, computes a proper interval model of a minimal proper interval completion of G.*

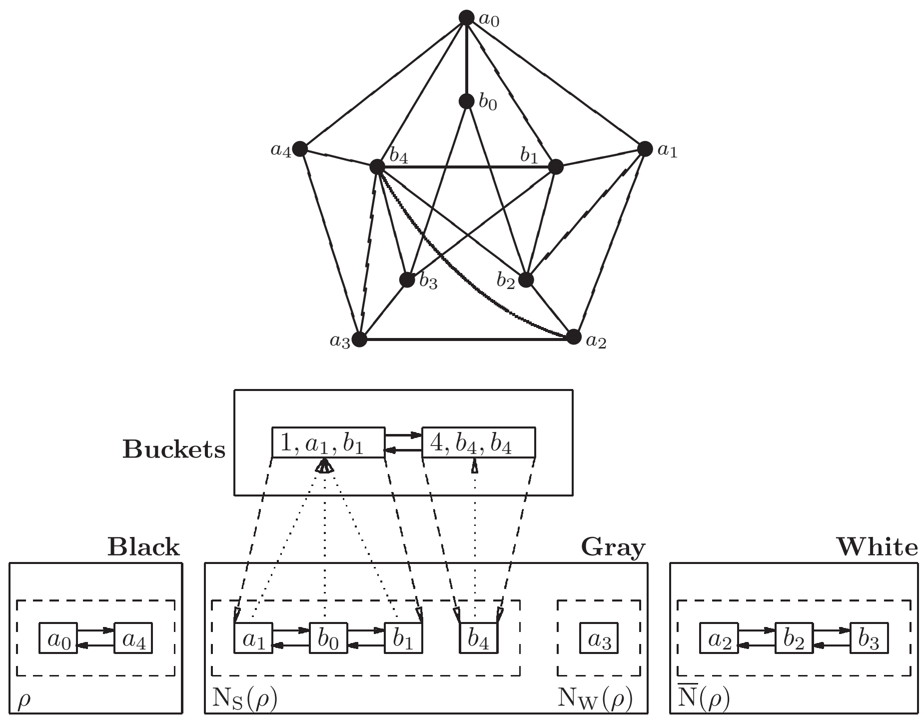
Proof. The ordering produced by the algorithm respects the conditions of Theorem 4. Indeed, it is sufficient to notice that the function **ChooseNextVertex** chooses a vertex in $\mathrm{N_S}(\rho)$ for the current prefix ρ, and moreover this vertex v is of minimum $\mathrm{d}_{\overline{\mathrm{N}}}(v)$. Since $\mathrm{d}_{\overline{\mathrm{N}}}(v)$ is the cardinality of $N(v) \cap \overline{\mathrm{N}}(\rho)$, the latter is inclusion-minimal among all the elements of $\mathrm{N_S}(\rho)$.

Let us discuss a linear time implementation of the algorithm. The choice of the first vertex can be done in linear time by Proposition 1. The main difficulty is that the function **ChooseNextVertex** must work in constant time. For this purpose, the queue Q will actually be a queue of sets $(N_{j_1}, N_{j_2}, \ldots, N_{j_k})$, where N_{j_p} is the set of neighbors of v_{j_p} added to the queue when processing v_{j_p} (empty sets are not enqueued). Hence $\mathrm{N_S}(\rho)$ is the first set in the queue, and vertices are dequeued from it.

In order to choose the vertex $v \in \mathrm{N_S}(\rho) = N_{j_1}$ of minimum $\mathrm{d}_{\overline{\mathrm{N}}}(v)$ in constant time, we need to sort $\mathrm{N_S}(\rho)$ by increasing $\mathrm{d}_{\overline{\mathrm{N}}}()$. Notice that the value of some $\mathrm{d}_{\overline{\mathrm{N}}}(z)$ might change during the algorithm, and we wish to update the value in constant time. We use a bucket sort with a special data structure (see [5, 6] for a detailed description of the data structure, the authors use it for partition refinement algorithms). The buckets are kept as a doubly chained list (instead of the usual array). Each bucket has its value (the $\mathrm{d}_{\overline{\mathrm{N}}}(u)$ for the elements of the bucket), and points towards the previous and next non-empty buckets, according to their values. The vertices of a bucket are kept in a doubly chained list, and each vertex points towards the bucket to which it belongs. An example is given in Figure 1, where the bucket of value 1 contains a_1, b_0, b_1 and the bucket 4 contains b_4.

When a set N_i becomes the first set of Q, we apply a classical bucket sort on the vertices of N_i. This sort costs $\mathcal{O}(|N_i| + \max\{\mathrm{d}_{\overline{\mathrm{N}}}(u) \mid u \in N_i\})$. Then we construct our data structure for the buckets, within the same running time. During the whole algorithm, this initialization of the buckets costs $\mathcal{O}(n+m)$, due to the fact that the sets N_i are pairwise disjoint.

During the algorithm, we decrement the value $\mathrm{d}_{\overline{\mathrm{N}}}(z)$ for some vertices z (see the two last **for** loops). If z is in the set $\mathrm{N_S}(\rho)$, we must update the buckets in constant time. Let B be the bucket containing z and B' be the previous bucket in the list of buckets. If the bucket B' corresponds to the value $\mathrm{d}_{\overline{\mathrm{N}}}(z) - 1$ (before decrementing it), we simply move z from B to B', and possibly remove B if it becomes empty. Otherwise, B' corresponds to a value strictly smaller than $\mathrm{d}_{\overline{\mathrm{N}}}(z) - 1$, we create a new bucket B'', of value $\mathrm{d}_{\overline{\mathrm{N}}}(z) - 1$, and add it to the list of buckets between B' and B. Thanks to our data structure, this operation can be done in linear time. Note that the total number of iterations of the two last **for** loops is at most



Algorithm MinimalProperIntervalCompletion

Input: a simple graph $G = (V, E)$;
Output: the proper interval model of a minimal proper interval completion of G;
Data structures:
    mark : each vertex is marked white (unprocessed), grey (being processed) or black (processed).
    Q is the queue of processed vertices.
    ρ is the current prefix (an ordering on the black vertices).
    $\mathrm{d}_{\overline{\mathrm{N}}}(v)$ is the number of white neighbors of the vertex v.
Function ChooseNextVertex : chooses a vertex v in the queue Q
    such that $v \in \mathrm{N_S}(\rho)$ and $\mathrm{d}_{\overline{\mathrm{N}}}(v)$ is minimum for this property.
Function IntervalModel : computes the interval model.
begin
compute a moplexian vertex v_1 and put $\rho := (v_1)$
mark all vertices as white
init Q with the neighbors of v_1 in G and mark these vertices as grey
mark v_1 as black
compute $\mathrm{d}_{\overline{\mathrm{N}}}(x)$ for all vertices x
for $i := 2$ to n do
    $v_i :=$ **ChooseNextVertex**()
    mark v_i as black
    $\rho := \rho \bullet v_i$
    compute the set N_i of white neighbors of v_i
    mark the elements of N_i as grey and add them to Q
    for each $y \in N_i$ // we update $\mathrm{d}_{\overline{\mathrm{N}}}()$
        for each neighbor z of y
            $\mathrm{d}_{\overline{\mathrm{N}}}(z) := \mathrm{d}_{\overline{\mathrm{N}}}(z) - 1$
$\sigma := \rho$
IntervalModel(σ)
end

Fig. 1. Algorithm Minimal Proper Interval Completion and data structure

```
Function IntervalModel

Input: σ = (v_1, ..., v_n) - a BFS ordering of a simple connected graph G;
Output: an interval model of the graph G(σ);
Data structures:
        r is the biggest index of a neighbor of a vertex considered so far.
        c is a counter for numbering the maximal cliques of G(σ).
        [v_{l_c} : v_{r_c}], 1 ≤ c ≤ j are maximal cliques of G(σ). (see Remark 1)
        v_{l_c}, v_{r_c}, 1 ≤ c ≤ j are marks on the leftmost and rightmost vertices of the maximal clique c.
        Q is a queue containing the numbers of maximal cliques that the current vertex belongs to.
begin
r := 1
c := 1
for i := 1 to n do
    if max{q | v_q ∈ N_G(v_i)} > r then
        r := max{q | v_q ∈ N_G(v_i)}
        mark v_i as v_{l_c}
        mark v_r as v_{r_c}
        increment c
for i := 1 to n do
    if v_i is marked as v_{l_c} then
        add c at the end of the queue Q
    assign to v_i the interval [First(Q) : Last(Q)]
    if v_i is marked as v_{r_c} then
        remove c from the beginning of the queue Q
im := the interval model
FixIntervalModel(σ, im)
end
```

Fig. 2. Algorithm Interval Model

$n + m$. Indeed, each vertex y becomes grey exactly once, thus each edge $\{y, z\}$ is visited at most twice.

The function **IntervalModel** (see Figure 2) constructs a clique path of $G(\sigma)$ like in Remark 1 and computes an interval model based on this clique path in linear time. Unfortunately the interval model obtained from the clique path is not directly a proper interval model, thus we have to mend it into a proper one. The task is accomplished in linear time by function **FixIntervalModel** (see Figure 3 and Theorem 6).

4.1 Mending the proper interval model

Let $\sigma = (v_1, \ldots, v_n)$ be a BFS ordering of a simple graph G. Let $(K_1, \ldots, K_j)$ be the clique path of $G(\sigma)$. Let $IM = \{[l_i : r_i] \mid 1 \leq i \leq n\}$, be the interval model of based on the clique path of $G(\sigma)$ (see 1. Inevitably, if only $G(\sigma)$ is not a clique, IM contains an interval strictly contained in another. Indeed, since K_1, K_2 are maximal cliques, there are vertices $v_a \in K_1 \setminus K_2$ and $v_b \in K_2 \setminus K_1$. Moreover, $G(\sigma)$ is connected, so there is a path $P_{a,b}$ between v_a and v_b. $K_1 \cap K_2$ separates v_a from v_b in $G(\sigma)$, so it contains a vertex v_c on $P_{a,b}$. Therefore, the interval $[l_a : r_a]$ is strictly contained in the interval $[l_c : r_c]$. Nevertheless we have the following:

Lemma 7. *Let $\sigma = (v_1, \ldots, v_n)$ be a BFS ordering of a simple graph G. Let $(K_1, \ldots, K_j)$ be the clique path of $G(\sigma)$. Let $IM = \{[l_i : r_i] \mid 1 \leq i \leq n\}$, be the interval model of based on the clique path of $G(\sigma)$. Then there are no vertices v_x, v_y with $l_x < l_y \leq r_y < r_x$.*

Algorithm FixIntervalModel

Input: $\sigma = (v_1, ..., v_n)$ - a BFS ordering of a simple connected graph G;
$[l_i : r_i]$, $1 \leq i \leq n$ - interval model based on the clique path of $G(\sigma)$;
(see Algorithm **IntervalModel**, Figure 2)
Output: a proper interval model of the graph $G(\sigma)$;
Data structures:
begin is the last clique that some of the vertices considered so far start at.
$count_b$ is the counter of vertices that start at *begin*.
end is the last clique that some of the vertices considered so far finish at.
$count_e$ is the counter of vertices that start at *end*.
$\epsilon := \frac{1}{2(n+1)}$ is the value used for enlarging.
begin
$begin := 1$, $count_b := 1$, $end := 1$, $count_e := 1$
for $i := 1$ **to** n **do**
  if $l_i = begin$ **then**
    $l_i := l_i - (n - count_b) * \epsilon$
    increment $count_b$
  else
    $begin := l_i$
    $count_b := 1$
  if $r_i = end$ **then**
    $r_i := r_i + count_e * \epsilon$
    increment $count_e$
  else
    $end := r_i$
    $count_e := 1$
end

Fig. 3. Algorithm Fix Interval Model

Proof. Suppose v_x, v_y are such vertices. Let K_a, K_b be the maximal cliques with $a = l_y - 1$, $b = r_y + 1$. Notice that $l_x \leq a < b \leq r_x$. Let $v_a \in K_a \setminus K_{l_y}$, $v_b \in K_b \setminus K_{r_y}$. They all exist, since $K_{l_x}, K_{l_y}, K_{r_x}$ are distinct maximal cliques of $G(\sigma)$. Moreover, $\{v_a, v_y, v_b\}$ is a stable set and all three vertices are adjacent to v_x. Therefore, together they form a claw, which yields a contradiction. □

So if an interval $[l : r]$ is strictly contained in $[l', r']$, then they either have a common start or a common end. In the first case, we want to slightly enlarge $[l : r]$ to the left, not changing the intersections with the other intervals. Moreover, the shorter the interval is the more we enlarge it, in order to have no more inclusion among intervals starting at l. In the other case, we enlarge $[l, r]$ to the right in the same manner.

Lemma 8. *Let $\sigma = (v_1, \ldots, v_n)$ be a BFS ordering of a simple graph G. Let $(K_1, \ldots, K_j)$ be the clique path of $G(\sigma)$. Let $IM = \{[l_i : r_i] \mid 1 \leq i \leq n\}$, be the interval model of based on the clique path of $G(\sigma)$. The vertices corresponding to groups of intervals that share the common start point are consecutive in σ. Moreover, they are sorted by inclusion with the shortest interval coming first in σ. The same holds for intervals that share the common end point, only that the shortest interval comes last in σ.*

Proof. Let IS_a be the intervals that start at a. That means the corresponding set of vertices VS_a is contained in K_a, the maximal clique of $G(\sigma)$, and in no maximal clique $K_c, c < a$. By construction of IM, VS_a are consecutive in σ. Indeed, any vertex v_y with $x < y < z, v_x \in$

$V_a, v_z \in V_a$ is contained in K_a and in no maximal clique $K_c, c < a$. Let $v_i, v_j \in VS_a$, $i < j$. By Remark 1, v_j is contained in every maximal clique of $G(\sigma)$ that v_i is. So they are sorted by inclusion of corresponding intervals. Clearly, the corresponding statement holds also for IE_b, the intervals that end at b. □

Notice that, instead of using σ to have the vertices sorted by inclusion of corresponding intervals, we could also do bucket sort in respect to the other ends.

Theorem 6. *The algorithm FixIntervalModel (see Figure 3) yields a proper interval model in linear time.*

Proof. For each group of vertices sharing the common start, the algorithm enlarges the corresponding intervals: the sooner the vertex appears the more. For each group of vertices sharing the common end, the proportion is reversed. Lemma 8 guarantees, that this is the right order. By Lemmas 7, all cases of inclusion are covered. The choice of ϵ ensures that we enlarge the intervals by strictly less than $1/2$ in each direction. Thus the intersections stay unchanged. Clearly, the complexity is $\mathcal{O}(n)$. □

5 Conclusions and perspectives

We presented a polynomial time algorithm computing a minimal proper interval completion of an arbitrary graph.

There are two very natural questions related to minimal proper interval completions that we leave open. The first would be to characterize all minimal proper interval completions, for example by describing all the orderings σ such that $G(\sigma)$ is a minimal proper interval completion of G. We point out that our algorithm cannot obtain any such ordering of the input graph. Indeed, if we consider the graph $K_{1,4}$, our algorithm chooses a simplicial vertex and completes the rest into a clique. A different minimal proper interval completion of the $K_{1,4}$ can be obtained by adding a matching to the independent set. For this particular example, we are able to construct all nice orderings, by a slightly different (and slower) technique. Roughly speaking, we can use a minimal separator S to split the graph into two parts (by partitioning the components of $G - S$ in two). We compute an ordering starting with the vertices of the minimal separator for one of the parts, then reverse it and use it as prefix to order the second part. It is tempting to ask whether this technique provides all possible nice completions.

The second question consists in extracting a minimal proper interval completion from some non-minimal proper interval completion H of G. The naive technique would consist in checking, for each edge $e \in E(H) \setminus E(G)$, if $H - e$ is a proper interval graph. Although this idea works for minimal triangulations and minimal split completions, in our case we have examples showing that it does not always yield a minimal proper interval completion.

References

1. A. Berry, J. P. Bordat, *Separability Generalizes Dirac's Theorem.* Discrete Applied Mathematics, 84(1-3): 43-53, 1998.
2. H. L. Bodlaender, *A Linear-Time Algorithm for Finding Tree-Decompositions of Small Treewidth.* SIAM Journal on Computing, 25(6):1305-1317, 1996.
3. L. Cai, *Fixed-Parameter Tractability of Graph Modification Problems for Hereditary Properties.* Information Processing Letters, 58(4):171-176, 1996.
4. M. C. Golumbic, *Algorithmic Graph Theory and Perfect Graphs.* Academic Press, 1980.

5. 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.
6. M. Habib, R. M. McConnell, C. Paul, L. Viennot, *Lex-BFS and partition refinement, with applications to transitive orientation, interval graph recognition and consecutive ones testing.* Theoretical Computer Science, 234(1-2): 59-84, 2000.
7. P. Heggernes, F. Mancini, *Minimal Split Completions of Graphs.* Proceedings of LATIN 2006, Lecture Notes in Computer Science, 3887:592-604, 2006.
8. 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.
9. 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.
10. D. Kratsch, J. Spinrad, *Minimal fill in $\mathcal{O}(n^{2.69})$ time.* To appear in Discrete Applied Mathematics.
11. 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.
12. 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.
13. 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.
14. B. S. Panda, S. K. Das, *A linear time recognition algorithm for proper interval graphs.* Information Processing Letters, 87(3): 153-161, 2003.
15. I. Rappaport, K. Suchan, I. Todinca, *Minimal proper interval completions.* Technical Report RR-2006-02, LIFO - University of Orléans, 2006.
http://www.univ-orleans.fr/SCIENCES/LIFO/prodsci/rapports/RR2006.htm.en.