An $\mathcal{O}(n^2)$ time Algorithm for the Minimal Permutation Completion Problem

Christophe Crespelle* Anthony Perez† Ioan Todinca‡

Abstract

We provide an $O(n^2)$ time algorithm computing a minimal permutation completion of an arbitrary graph $G = (V, E)$, i.e., a permutation graph $H = (V, F)$ on the same vertex set, such that $E \subseteq F$ and F is inclusion-minimal among all possibilities.

1 Introduction

In graph modification problems, we are given an arbitrary input graph G and the goal is to transform it, using a small number of "modifications", into a graph satisfying some property Π. Typically, modifications consist in adding and/or removing edges and/or vertices. Here we consider the case where we are only allowed to add edges to the input graph $G = (V, E)$, transforming it into a super-graph $H = (V, F)$, such that H belongs to some required class of graphs.

Probably the most famous problem of this kind is MINIMUM FILL-IN, where the goal is to add as few edges as possible in order to obtain a *chordal* graph H. A graph is chordal if it has no induced cycles with four or more vertices. The problem being NP-hard [15], it triggered the attention to a simpler one, where we are only required to compute an inclusion-minimal chordal supergraph H of G. Such a graph is called a *minimal triangulation* of G, and the MINIMAL TRIANGULATION problem has been known to be polynomial since 1976 [14, 12]. The problem can be solved in time $O(nm)$ [14], and when the graph is dense the current best algorithm is the one of Heggernes et al. [7]. A detailed survey on this problem is provided in [5].

Minimal completions into other graph classes have been intensively studied. There are polynomial algorithms computing minimal completions into *interval graphs* [11, 4], *proper interval graphs* [13], *split graphs* [8], *cographs* [9] and *comparability graphs* [6]. The *minimum* versions of these problems are NP-complete (see, e.g., the thesis of Mancini [10] for further discussion and references).

In this paper we consider the problem of minimal completions into *permutation* graphs. A graph is a permutation graph if we can assign to each vertex a segment, all segments having an

*Université Claude Bernard Lyon 1, DANTE/INRIA, LIP UMR CNRS 5668, ENS de Lyon, Université de Lyon – christophe.crespelle@inria.fr

†Univ. Orléans, INSA Centre Val de Loire, LIFO EA 4022, FR-45067 Orléans, France – anthony.perez@univ-orleans.fr

‡Univ. Orléans, INSA Centre Val de Loire, LIFO EA 4022, FR-45067 Orléans, France – ioan.todinca@univ-orleans.fr

endpoint on a "top" line and the other one on a parallel "bottom" line, such that two vertices are adjacent if and only if the two corresponding segments intersect. Such a representation is called a *permutation model* of the graph. We give an $O(n^2)$ algorithm computing a minimal permutation completion of an arbitrary graph. This is, to the best of our knowledge, the first polynomial algorithm for the problem. Let us point out that computing a permutation completion with a minimum number of edges is NP-hard [2].

Our result is based on a vertex-incremental approach, also used for other types of completions. More specifically, we take the vertices of the input graph one by one, in an arbitrary order, and at each step we add the new vertex x_i to the previously computed minimal permutation completion H_{i-1}. The new minimal permutation completion H_i is obtained by only adding edges between x_i and the rest of the graph. Very informally, if we are given a permutation model of H_{i-1}, we need to insert the segment of x_i in a minimal way. In Section 3 we consider the case when H_{i-1} has a unique permutation model and we provide an efficient computation, in $O(n)$ time, of such an insertion position. Somehow surprisingly, even this task is non-trivial (the similar algorithm is very simple in the case of minimal interval completions). Then we need to take into account (Section 4) the fact that H_{i-1} may have many different models, fortunately they are all encoded in its *modular decomposition*. Eventually, we compute the modular decomposition of the new completion H_i. This is done thanks to the algorithm of Crespelle and Paul [3], which incrementally maintains the modular decomposition of permutation graphs.

2 Preliminaries

Every graph $G = (V, E)$ considered here will be finite, undirected, loopless and simple. We denote $V(G)$ the set of vertices of G and $n = |V(G)|$. The edge between vertices x and y will arbitrarily be denoted either xy or yx. For a subset $S \subseteq V$ of vertices, we denote by $G[S]$ the subgraph of G induced by S.

A graph $G = (V, E)$ is a *permutation graph* if and only if it admits a *permutation model* (π_1, π_2), i.e. two one-to-one mappings $\pi_1, \pi_2 : V \to \{1, \ldots, n\}$ such that two vertices x and y are adjacent in G if and only if $(\pi_1(x) - \pi_1(y))(\pi_2(x) - \pi_2(y)) < 0$. An equivalent geometric definition of a permutation model for G is to associate to each vertex x a segment, each segment having one endpoint on a *top line* and the other on a parallel *bottom line* and all endpoints being pairwise distinct, such that two vertices x and y are adjacent if and only if the corresponding segments intersect. The correspondence between these two definitions is that the order in which appear the endpoints on the top (resp. bottom) line of the permutation model is the order defined by mapping π_1 (resp. π_2). We equally use these two visions in the rest of the article, depending on which one is more convenient for our purpose.

Note that a permutation model can be encoded by storing mappings π_1 and π_2 as arrays, which uses $O(n)$ space under the usual assumption that integers smaller then n are stored in constant space. Then, the condition for x and y to be adjacent can be tested in $O(1)$ time by two comparisons of integers.

Let now $G = (V, E)$ be an arbitrary graph. A *permutation completion* of G is a permutation graph $H = (V, F)$, on the same vertex set, such that $E \subseteq F$. If, moreover, set F is inclusion-minimal under these constraints, we say that H is a *minimal permutation completion* of G. In this paper we deal only with permutation completions, thus we will sometimes simply refer to

these completions as (minimal) completions, omitting the term "permutation".

Note that every graph G has a permutation completion: one can simply add all the missing edges to G and observe that the complete graph is a permutation graph. Permutation graphs are also *hereditary*, i.e., an induced subgraph of a permutation graph is also a permutation graph (it is sufficient to restrict an permutation model of the original graph to the vertices of the induced subgraph).

Our approach for computing a minimal completion of an arbitrary graph G is incremental, in the sense that we take the vertices of G one by one, in an arbitrary order $(x_1, \ldots, x_n)$, and at step i we compute a minimal permutation completion H_i of $G_i = G[\{x_1, \ldots, x_i\}]$ from a minimal permutation completion H_{i-1} of G_{i-1}, by adding only edges incident to x_i. This is possible thanks to the following observation that is general to all hereditary graph classes that are also stable by addition of a universal vertex, in particular for permutation graphs.

Lemma 1 (see, e.g., [11]) *Let G be an arbitrary graph and let H be a minimal permutation completion of G. Consider a new graph $G' = G + x$, obtained by adding to G a new vertex x adjacent to an arbitrary set $N(x)$ of vertices of G. There is a minimal permutation completion H' of G' such that $H' - x = H$.*

For any subset $W \subseteq V(G)$ of vertices, we say that we *fill* W in H' if we make all the vertices of $W \setminus N(x)$ adjacent to x in the completion H' of $G + x$.

The new problem. From now on, we consider the following problem, with slightly modified notations.

Let G be a permutation graph, and let $G + x$ be the graph obtained by adding to G a new vertex x adjacent to some set $N(x)$ of vertices of G. Our goal is to compute a minimal permutation completion H of $G + x$ such that $H - x = G$.

We will solve this problem in $O(n)$ time, where n is the number of vertices of G. As we shall see, graph G will be given together with its *modular decomposition*, which is known to encode all its possible permutation models. Our algorithm will compute the set $N'(x) \supseteq N(x)$ of neighbours of x in graph H, and will update this data structure for the completion H, thanks to the algorithm of [3].

3 Minimal completion respecting a permutation model

Definition 1 (Minimal completion respecting a permutation model) *Given a permutation model (π_1, π_2) of a permutation graph G and a vertex x to insert in G, a permutation completion H of $G + x$* respects *(π_1, π_2) if there exists a permutation model of H such that removing x from it results in (π_1, π_2). Moreover, if H is inclusion minimal among such completions, then H is called a* minimal permutation completion respecting *(π_1, π_2).*

The goal of this section is an $O(n)$ algorithm computing a minimal permutation completion respecting a given permutation model of a graph G.

Theorem 1 *Given a permutation model (π_1, π_2) of a permutation graph G and a vertex x to insert in G together with its neighbourhood $N(x)$, a minimal permutation completion H of $G + x$ respecting (π_1, π_2) can be computed in $\mathcal{O}(n)$ time. The output is a permutation model of H.*

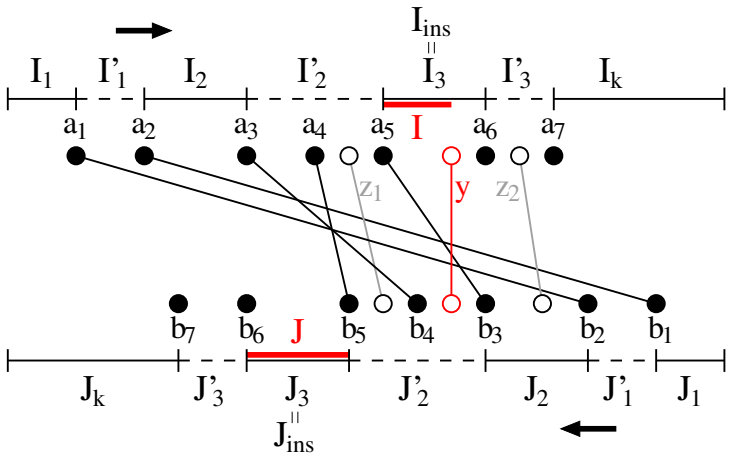

Figure 1: Some definitions and notations.

This does not provide in general a minimal completion of G as a permutation graph may admit many different permutation models. Nevertheless, such a completion is minimal when graph G is prime (in which case it admits a unique permutation model, up to symmetries) and this is the only case where we will use this approach in our global algorithm. However, the algorithm we design in this section is not specific to prime graphs and we present it in the general setting.

In order to obtain a permutation model of H respecting (π_1, π_2), we extend the definition of functions π_1 and π_2, initially defined from $V(G)$ to $\{1, \ldots, n\}$, to vertex x as well, in the following way.

Definition 2 (Insertion position) *An* insertion position *of x in the permutation model (π_1, π_2) is a couple $(\pi_1(x), \pi_2(x)) = (i + 0.5, j + 0.5)$ for some integers $i, j \in \{0, \ldots, n\}$.*

Observe that a necessary and sufficient condition for insertion position $(\pi_1(x), \pi_2(x))$ to give a permutation completion of $G + x$ is that $\{z \in N(x) \mid \pi_1(z) < \pi_1(x)\} = \{t \in N(x) \mid \pi_2(t) > \pi_2(x)\}$. We now characterise all such insertion positions of x by a family of *slots* (I_j, J_j), i.e., couples of intervals, such that a position $Pos = (p_1, p_2)$ can be assigned to x in order to obtain a completion of $G + x$ if and only if (p_1, p_2) is contained in one of the slots (I_j, J_j). Moreover, the intervals I_j (resp. J_j) will be pairwise disjoint. To this purpose, we adopt the following notations, see their illustration on Figure 1.

Let $l = |N(x)|$, $a_1, a_2, \ldots, a_l$ denote the neighbours of x in **increasing** order in π_1, i.e. $\pi_1(a_1) < \pi_1(a_2) < \cdots < \pi_1(a_l)$, and $b_1, b_2, \ldots, b_l$ denote the neighbours of x in **decreasing** order in π_2, i.e. $\pi_2(b_l) < \cdots < \pi_2(b_2) < \pi_2(b_1)$. For $1 \leq i \leq l - 1$, we also denote $A_i = \{a_1, \ldots, a_i\}$ and $B_i = \{b_1, \ldots, b_i\}$. We define $Valid = \{i \in \{1, \ldots, l - 1\} \mid A_i = B_i\}$. Let $k = |Valid|+2$ and we denote the elements of $Valid$ by $v_2, v_3, \ldots, v_{k-1}$ in increasing order. Then, for any $j \in \{2, \ldots, k - 1\}$, let $I_j = [\pi_1(a_{v_j}), \pi_1(a_{v_j+1})]$ and $J_j = [\pi_2(b_{v_j+1}), \pi_2(b_{v_j})]$ (cf. Figure 1). We also denote $I_1 = [0, \pi_1(a_1)]$, $J_1 = [\pi_2(b_1), n + 1]$, $I_k = [\pi_1(a_l), n + 1]$ and $J_k = [0, \pi_2(b_l)]$. Note that the intervals I_j, $1 \leq j \leq k$, are pairwise disjoint and numbered from left to right in π_1, while the intervals J_j are pairwise disjoint and numbered from right to left in π_2.

With these notations, the insertion positions $(\pi_1(x), \pi_2(x))$ that define a completion H of $G + x$ (i.e., such that x is at least adjacent in H to all the vertices of $N(x)$) are exactly the insertion positions such that $\pi_1(x) \in I_i$ and $\pi_2(x) \in J_i$, for some $i \in \{1, \ldots, k\}$. Our next goal is to choose, among all slots (I_i, J_i), an insertion slot (I_{ins}, J_{ins}) that will provide an optimal insertion position (Definition 3). Then, through a sequence of lemmas, we prove that this slot contains indeed a position providing a minimal permutation completion compatible with our permutation model (Lemma 3).

Given an interval I, let $l(I)$ (resp. $r(I)$) denote the left endpoint (resp. right endpoint) of I. For any $i \in \{1, \ldots, k-1\}$, we define $I'_i = [r(I_i), l(I_{i+1})]$ (cf. Figure 1), and $I'_k = \varnothing$. Symmetrically, in π_2, we denote $J'_i = [r(J_{i+1}), l(J_i)]$, and $J'_k = \varnothing$. Finally, we denote $\hat{I}_i = I_i \cup I'_i$ and $\hat{J}_i = J_i \cup J'_i$.

Definition 3 (Forced, forwarding, insertion slot and vertex y) *A vertex z is* forced *if $z \in N(x)$ or there exists $i \in \{1, \ldots, k\}$ such that $\pi_1(z) \in I'_i$ and $\pi_2(z) \in J'_i$.*

We say that $i \in \{1, \ldots, k\}$ is forwarding *if all vertices y such that $\pi_1(y) \in \bigcup_{1 \leq j \leq i} \hat{I}_j$ and $\pi_2(y) \in \bigcup_{1 \leq j \leq i} \hat{J}_j$ are forced. The* insertion slot (I_{ins}, J_{ins}) *is defined by $ins = \min\{i \in \{1, \ldots, k\} \mid i$ is not forwarding$\}$ and we denote by y a non-forced vertex that makes ins not forwarding.*

First, note that forced vertices are adjacent to x in any completion of $G + x$ respecting (π_1, π_2). Of course, it may happen that all i are forwarding. In this case, x is adjacent to all the vertices of G in any completion respecting (π_1, π_2). Consequently, the unique minimal such completion is easily obtained by inserting x in any arbitrary slot (I_i, J_i), with $i \in \{i, \ldots, k\}$. In the following, we do not consider this case anymore and we assume that there exists a slot i which is not forwarding. Then, the insertion slot ins and the vertex y are well defined. Observe that, by minimality of ins, we either have $\pi_1(y) \in \hat{I}_{ins}$ and $\pi_2(y) \in \bigcup_{1 \leq j \leq ins} \hat{J}_j$, or $\pi_2(y) \in \hat{J}_{ins}$ and $\pi_1(y) \in \bigcup_{1 \leq j \leq ins} \hat{I}_j$.

We make some crucial observations about the slot (I_{ins}, J_{ins}), that will allow to prove that one can find a minimal completion by inserting x in this slot.

Lemma 2 *There is a completion obtained by inserting x in the slot (I_{ins}, J_{ins}) such that x is not adjacent to y in this completion.*

Moreover, this holds exactly for insertion positions in a sub-slot $(I, J) \subseteq (I_{ins}, J_{ins})$ defined as follows. Denote $I_y^- = [0, \pi_1(y)]$, $I_y^+ = [\pi_1(y), n+1]$, $J_y^- = [0, \pi_2(y)]$ and $J_y^+ = [\pi_2(y), n+1]$. We distinguish two cases:

1. *if $\pi_1(y) \in I_{ins}$ or $\pi_2(y) \in J_{ins}$ but not both, then there exists a unique $\alpha \in \{-, +\}$ s.t. $I_{ins} \cap I_y^\alpha \neq \varnothing$ and $J_{ins} \cap J_y^\alpha \neq \varnothing$, and we set $(I, J) = (I_{ins} \cap I_y^\alpha, J_{ins} \cap J_y^\alpha)$;*
2. *otherwise, $(\pi_1(y), \pi_2(y)) \in (I_{ins}, J_{ins})$ and then there are two suitable slots $(I, J) = (I_{ins} \cap I_y^-, J_{ins} \cap J_y^-)$ and $(I, J) = (I_{ins} \cap I_y^+, J_{ins} \cap J_y^+)$.*

Lemma 3 *Let C be a completion obtained by inserting x in the slot (I_{ins}, J_{ins}) such that x is not adjacent to y in the completion, and C is minimal among such completions. Then, C is a minimal completion of $G + x$ respecting (π_1, π_2).*

Sketch of proof. Such a completion C exists by Lemma 2. Remind that all the completions respecting (π_1, π_2) are obtained by inserting x in some slot (I_i, J_i), with $1 \leq i \leq k$. Moreover, if $i > ins$ it is straightforward to see that the completions obtained by inserting x in (I_i, J_i) make x adjacent to y and are therefore not subgraphs of completion C. Now, let D be a completion obtained by inserting x in (I_i, J_i) with $i < ins$. Consider the completion C' obtained by inserting x at position $(l(I_{ins}) + 0.5, r(J_{ins}) - 0.5)$ in (I_{ins}, J_{ins}). The fact that $ins - 1$ is forwarding implies that C' is contained in D. If x is adjacent to y in C', clearly D is not a subgraph of C. If x is not adjacent to y in C', then C' is not a strict subgraph of C (by minimality of C), thus D is not a strict subgraph of C. □

All insertion positions in (I_{ins}, J_{ins}) avoiding to make x adjacent to y are characterised by Lemma 2 as the insertion positions in a slot (I, J) of subintervals of respectively I_{ins} and J_{ins} (actually two couples in the second case of the lemma). Our next goal is to compute an optimal insertion position contained in a given slot (I, J), Lemma 4 below. We need the following definition.

Definition 4 ((Rightmost) left-stable insertion position) *Let (I, J) be two intervals with endpoints in $\{0, \ldots, n+1\}$. An insertion position $Pos = (p_1, p_2)$ for x in (I, J) is* left-stable *if all vertices z such that $l(I) < \pi_1(z) < p_1$ (resp. $l(J) < \pi_2(z) < p_2$) satisfy $\pi_2(z) < p_2$ (resp. $\pi_1(z) < p_1$).*
Moreover, there is a unique left-stable insertion position Pos, called the rightmost *one, such that there is no left-stable insertion position $Pos' = (p'_1, p'_2)$ strictly on the right of Pos in (I, J), i.e. different from Pos and having $p'_1 \geq p_1$ and $p'_2 \geq p_2$.*

The uniqueness of the rightmost left-stable insertion position comes from the fact that two left-stable insertion positions (p_1, p_2) and (p'_1, p'_2) that cross each other define another left-stable insertion position $(\max\{p_1, p'_1\}, \max\{p_2, p'_2\})$ which is strictly on the right of both of them.

Lemma 4 *Let $Pos = (p_1, p_2)$ be the rightmost left-stable insertion position in (I, J). Then, the completion obtained by inserting x at position Pos is minimal among completions obtained by insertion of x in (I, J).*

Sketch of proof. Let Pos' be another insertion position in (I, J) and let C' (resp. C) be the completion obtained from Pos' (resp. Pos). We distinguish three cases to show that C' is not strictly contained in C. Firstly, if the segment of Pos' crosses the one of Pos, then, because of the left-stability of Pos, in the order where the endpoint of Pos is on the right side of the one of Pos', there is one vertex z whose segment crosses Pos' and not Pos. Then, C' is not contained in C. Now, if Pos' is strictly on the left of Pos, the left-stability of Pos implies that there is no vertex with one endpoint on the right of Pos and one between Pos and Pos'. Then, C is contained in C' and the conclusion follows. Finally, if Pos' is strictly on the right of Pos, then, by definition of Pos, Pos' is not left-stable. And since Pos is left-stable, there exists one vertex with one endpoint between Pos and Pos' and the other one on the right side of Pos', which implies that C' is not contained in C. □

The algorithm. Our algorithm is in two steps. The first step determines the insertion slot (I_{ins}, J_{ins}). To that purpose we first compute the slots (I_i, J_i) defined above, in $O(n)$ time as shown in [3]. Then, by increasing i, we scan the vertices having one endpoint in the slot (I_i, J_i)

until we find one non-forced vertex y, which can be tested in $O(1)$ time. We then have $ins = i$ and a suitable vertex y as in Definition 3. This takes $O(n)$ time.

The second step inserts x in (I_{ins}, J_{ins}). By Lemma 2, either there is a unique slot (I, J) that contains the optimal insertion position, or there are two possible slots (I_r, J_r) and (I_l, J_l) such that the first one is on the right. In the first case, we find (by an algorithm described a little below) the rightmost left-stable insertion position in (I, J), and it is optimal by Lemma 4. In the second case, we compute the rightmost left-stable insertion positions Pos_r and Pos_l in each of these slots (I_r, J_r) and (I_l, J_l). If the completion C_r obtained from Pos_r is included in the one C_l obtained from Pos_l, then Lemma 4 ensures that C_r is minimal among the completions respecting (π_1, π_2) and the algorithm returns Pos_r. Otherwise (when C_r is not contained in C_l), the algorithm returns position Pos_l. The correctness of this choice comes from the fact that if there exists a completion C' obtained by insertion of x in (I_r, J_r) that is contained in C_l, one can show that C_r is also necessarily contained in C_l. Note that inclusion of C_r into C_l can be easily tested in $O(n)$ time by scanning one order of the permutation model.

It remains to show how to compute the rightmost left-stable insertion position in a slot (I, J) in $\mathcal{O}(n)$ time. We start from the left-stable position $(p_1, p_2) = (l(I) + 0.5, l(J) + 0.5)$ and while it preserves left-stability, we increment p_1 (resp. p_2) by one. When it is no longer possible, the vertices $nextR(p_1)$ and $nextR(p_2)$ that are immediately on the right of p_1 and p_2 respectively satisfy $\pi_2(nextR(p_1)) > p_2$ and $\pi_1(nextR(p_2)) > p_1$. We then start a search for the leftmost left-stable position which is strictly on the right of (p_1, p_2) in I, if it exists. To that purpose we set $b_2 \leftarrow \pi_2(nextR(p_1))$ and $b_1 \leftarrow \pi_1(nextR(p_2))$, and we increment both p_1 and p_2. We then continue to scan π_1 (resp. π_2) by iteratively incrementing p_1 (resp. p_2), without going beyond b_1 (resp. b_2), and for every vertex z encountered in π_1 (resp. π_2) we set $b_2 \leftarrow max\{b_2, \pi_2(z)\}$ (resp. $b_1 \leftarrow max\{b_1, \pi_1(z)\}$). We proceed in this way simultaneously for p_1 and p_2 in an asynchronous manner until we get both $p_1 = b_1 - 0.5$ and $p_2 = b_2 - 0.5$, when no further increment is possible. At this stage, if (p_1, p_2) is no longer in (I, J) then the algorithm returns the previous left-stable position, say (p_1^{prev}, p_2^{prev}), and stops. Otherwise, (p_1, p_2) is the leftmost left-stable position on the right of (p_1^{prev}, p_2^{prev}) that we were looking for. Then, the algorithm continues to look for other left-stable positions on the right of current (p_1, p_2) by starting again at the beginning of this paragraph. When this process ends, we get the rightmost left-stable position in (I, J). As every treatment during the scan of π_1 and π_2 takes constant time, the overall complexity is $O(n)$. This achieves the proof of Theorem 1.

4 General minimal completion of $G + x$

For our general algorithm computing a minimal permutation completion of $G + x$, we consider the modular decomposition of G . For each node u of the modular decomposition tree T, we denote by $V[u]$ the subset of vertices of G appearing in the subtree rooted in u, and by $G[u]$ the subgraph of G induced by these vertices. We will denote by $G[u] + x$ the graph obtained from $G[u]$ by adding x with neighbourhood $N(x) \cap V[u]$.

A subset $W \subseteq V(G)$ of vertices is said to be *hit* in $G+x$ (resp. in a completion H of $G+x$) if it intersects the neighbourhood $N(x)$ of x in $G + x$ (resp. the neighbourhood $N'(x)$ of x in H). If W is contained in $N(x)$ (resp. $N'(x)$), we say that W is *full* in $G + x$ (resp. H). If W is hit but not full, we say it is *mixed*. We also say that a node u of the modular decomposition tree T of G is hit, full or mixed according to the status of its associated set of vertices $V[u]$. When

we omit to precise it, the graph referred to in these notions is $G + x$. Observe that the set of hit nodes of T can be computed in $\mathcal{O}(n)$ time by a bottom-up parsing of T.

For each node u of T, $P_u = (V_u, E_u)$ denotes the corresponding quotient graph . If u is a *prime* node, P_u is prime, and it is stored together with a permutation model. If u is a *series* or *parallel* node, then graph P_u is a complete graph, respectively an independent set.

In this section, it is more convenient to work with the geometric version of a permutation model where each vertex is represented by a segment, all segments have an extremity on the *top line* of the model and another one on the *bottom* line. By [1], any permutation model of $G[u]$ is obtained from a model of P_u by *expanding each segment corresponding to a vertex v_i of P_u into a model of $G[v_i]$* as follows. The segment of v_i is enlarged into a parallelogram by enlarging its top (resp. bottom) endpoint into a top (resp. bottom) edge, lying on the top (resp. bottom) line of the permutation model. These expansions are such that the top edges (resp. bottom edges) of the parallelograms are pairwise disjoint, and they appear on the top (resp. bottom) line in the same order as the endpoints of the segments. Therefore, for any pair of vertices v_i and v_j of P_u, the corresponding parallelograms intersect if and only if v_iv_j is an edge of P_u. Now, for each v_i (recall that v_i is a child of u in the modular decomposition tree), we insert a model of $G[v_i]$ into the parallelogram of v_i.

Definition 5 (Contracted graph $P_u + x$) *The* contracted graph $P_u + x$ *is the graph obtained from P_u by adding the vertex x with neighbourhood $N_{P_u+x}(x) = \{v_i \in V_u \mid v_i$ is hit$\}$.*

Let H_u be a minimal permutation completion of $P_u + x$. In order to obtain a permutation completion of $G[u] + x$, we could consider each node v_i adjacent to x in H_u and fill the set $V[v_i]$ (i.e., make all vertices of $V[v_i]$ adjacent to x). It is not hard to see that this construction yields a permutation completion of $G[u] + x$. Indeed, take a permutation model H_u^{mod} of H_u, and expand, for each node v_j of P_u, the segment of v_j into a parallelogram (the segment of x remains unchanged). By inserting a model of $G[v_j]$ into the parallelogram of v_j, we obtain a model for the completion of $G[u] + x$.

Such a completion is not necessarily minimal, because in the construction above we can sometimes "shift" the top and/or bottom endpoint of segment x inside a parallelogram and save some edges in the completion, by avoiding to have the segment of x cross some of the segments inside this parallelogram. As we shall see, there are at most two hit children v_i of u for which we do not fill $V[v_i]$. Depending on the cases, for such a child v_i, either we can use any minimal permutation completion of $G[v_i]$, or we have to use one which is minimal among those satisfying an additional constraint, which we call an *external-minimal permutation completion* (see Definition 6 below). This is the reason why our algorithm actually computes not just one but two permutation completions of $G[u] + x$ for each node u of the modular decomposition tree: one minimal permutation completion and one external-minimal permutation completion.

Definition 6 (External-minimal permutation completion) *A permutation completion H of $G + x$ is said to be an* external permutation completion *if H has a permutation model such that the bottom endpoint of segment x is the leftmost among all bottom endpoints of the segments of the model.*
If, moreover, H is minimal among all external completions, then H is called an external-minimal permutation completion*.*

Since any model can be reversed from left to right or upside-down, in the definition above we could replace "leftmost" by "rightmost" and "bottom" by "top".

The notion of *representation* defined below is central in the rest of the paper. We use it to construct a minimal completion of $G[u]+x$ from a minimal completion of P_u+x and minimal and external-minimal completions of $G[v_i]+x$.

Definition 7 (Representation) *Let H_u be a permutation completion of P_u+x. A* representation *of H_u is a triplet $(H_u^{mod}, v_{top}, v_{bot})$ or a couple (H_u^{mod}, v_{top}) or a single element H_u^{mod} such that:*

- *H_u^{mod} is a permutation model of H_u, and*
- *v_{top} (resp. v_{bot}) is a vertex of P_u that is adjacent to x in H_u and such that the top (resp. bottom) endpoint of segment v_{top} (resp. v_{bot}) is next to the top (resp. bottom) endpoint of segment x in H_u^{mod}.*

Observe that we might have the situation that both v_{top} and v_{bot} exist and are equal. The following definition shows how to use a representation of H_u in order to get a permutation completion of $G[u]+x$.

Definition 8 (Permutation completion resulting from a representation) *Let H_u be a permutation completion of P_u+x. Consider a representation $\mathcal{R}$ of H_u, with $\mathcal{R}=(H_u, v_{top}, v_{bot})$ or $\mathcal{R}=(H_u, v_{top})$ or $\mathcal{R}=H_u$, and assume that we are given, for each node $v_\alpha \in \{v_{top}, v_{bot}\}$, the neighbourhood N'_α (resp. N^{ext}_α) of x in a minimal (resp. an external-minimal) permutation completion of $G[v_\alpha]+x$.*
We construct a vertex set N' (initially empty) as follows. For each $v_i \in V_u$ adjacent to x in H_u and distinct from v_{top} and v_{bot}, we add $V[v_i]$ to N'. Then, if at least one of v_{top} and v_{bot} exists, we distinguish two cases:

1. *if both v_{top} and v_{bot} exist and $v_{top}=v_{bot}$, we add N'_{top} to N',*
2. *otherwise, i.e. if v_{bot} does not exist or if both v_{top} and v_{bot} exist and are not equal, for each $v_\alpha \in \{v_{top}, v_{bot}\}$,we add N^{ext}_α to N'.*

The permutation completion of $G[u]+x$ resulting from the representation $\mathcal{R}$ of H_u *is the one obtained from $G[u]+x$ by filling N'.*

The above definition is correct as filling N' indeed results in a permutation completion of $G+x$.

Finally, we refine the notion of representation for that of an *optimal representation*, based on a minimal permutation completion of P_u+x. The supplementary conditions required for a representation to be optimal ensure that the resulting permutation completion is minimal (Theorem 2 below).

Definition 9 (Optimal representation) *Let H_u be a* minimal *permutation completion of P_u+x. For each node v_i of P_u, let N'_i be the neighbourhood of x in a* minimal *completion of $G[v_i]+x$, and let N^{ext}_i be the neighbourhood of x in an* external-minimal *completion of $G[v_i]+x$.*

1. *If there is a representation $\mathcal{R}_1=(H_u^{mod}, v_{top}, v_{bot})$ of H_u such that v_{top} and v_{bot} are different and $N^{ext}_{top} \subsetneq V[v_{top}]$ and $N^{ext}_{bot} \subsetneq V[v_{bot}]$, we say that H_u is* of the first type *and $\mathcal{R}_1$ is an* optimal representation *of H_u.*

2. *If H_u is not of the first type, but there exists a representation $\mathcal{R}_2 = (H_u^{mod}, v_{top}, v_{bot})$ of H_u such that $v_{top} = v_{bot}$ and $N'_{top} \subsetneq V[v_{top}]$, we say that H_u is* of the second type *and $\mathcal{R}_2$ is an* optimal representation *of H_u.*

3. *If H_u is not of the first or second type, but there exists a representation $\mathcal{R}_3 = (H_u^{mod}, v_{top})$ such that $N_{top}^{ext} \subsetneq V[v_{top}]$, we say that H_u is* of the third type *and $\mathcal{R}_3$ is an* optimal representation *of H_u.*

4. *If none of the above holds, we say that H_u is of the fourth type, and the fourth-type optimal representation is simply H_u^{mod}, where H_u^{mod} is any model of H_u.*

We have a similar definition for producing an external-minimal completion of $G[u]+x$. We now prove the combinatorial theorem which constitutes the base of our algorithm for minimal permutation completion.

Theorem 2 *Let H_u be a minimal (resp. external-minimal) permutation completion of $P_u + x$ and let $\mathcal{R}$ be an optimal (resp. external-optimal) representation of H_u. Then, the permutation completion H of $G[u] + x$ resulting from $\mathcal{R}$ is minimal (resp. external-minimal).*

Sketch of proof. For lack of space, we only give a flavour of the proof in the case where H_u is a minimal completion of $P_u + x$ and H_u is of the first type . Assume for contradiction that there is some permutation completion H' of $G[u] + x$ such that H' is a strict subgraph of H. Apply the following transformation to a permutation model of H': keep the segment of x and for each node v_i in P_u keep only one vertex of $V[v_i]$ and choose it adjacent to x in H' if there exists one such vertex in $V[v_i]$. The model obtained is the one of a permutation completion H'_u of $P_u + x$. Applying the same process to a permutation model of H results in a model H_u^{mod} of H_u defining an optimal representation of H_u of the first type. Moreover, one can show that because H' is a subgraph of H, then H'_u is also a subgraph of H_u. Therefore, $H'_u = H_u$ by minimality of H_u. As a consequence, for every node v_i such that $V[v_i]$ is hit in H (hence v_i is adjacent to x in H_u) the set $V[v_i]$ is also hit in H' (because v_i is adjacent to x in H'_u).

Let now y be a vertex that is adjacent to x in H but not in H'. Let v_m be the node of P_u such that $y \in V[v_m]$, necessarily v_m is mixed in H'. Observe that, by construction (see Definition 8), since H_u is of the first type, then $V[v_{top}]$ and $V[v_{bot}]$ are mixed in H, and so in H' from what precedes. Also observe that in any completion of $G[u] + x$, the only children v_i of u such that $V[v_i]$ is mixed are those whose parallelogram contains at least one endpoint of x. Then, in any permutation model $H^{mod'}$ of H', the parallelogram of v_m contains one endpoint of x and only one, because the parallelograms of both v_{top} and v_{bot} also contain one endpoint of x and at least one of v_{top} and v_{bot} is different from v_m (v_{top} and v_{bot} are distinct since H_u is of the first type). Consequently, restricting $H^{mod'}$ to $V[v_m] \cup \{x\}$ shows that $H'[V[v_m] \cup \{x\}]$ is an external completion of $G[v_m] + x$. Moreover, by construction (see Definition 8), $H[V[v_m] \cup \{x\}]$ is an external-minimal completion of $G[v_m] + x$, contradicting the fact that $H'[V[v_m] \cup \{x\}]$ is a strict subgraph of $H[V[v_m] \cup \{x\}]$. This proves the theorem for the case where H_u is of the first type. □

The algorithm. Our $\mathcal{O}(n)$-time algorithm computing a minimal permutation completion of $G+x$ proceeds by a bottom-up computation of two functions, $MinIns(u)$ and $MinInsExt(u)$, for each node u. Function $MinIns(u)$ computes a minimal permutation completion of $G[u] +$

x (with the corresponding neighbourhood $N(x) \cap V[u]$). It returns the neighbourhood N'_u of x in this completion. Function $MinInsExt(u)$ computes a minimal-external permutation completion of $G[u]+x$, i.e., it returns the neighbourhood N_u^{ext} of x in such a completion. These functions will only be called on hit nodes, from bottom to top. Therefore we can assume that, when we treat a node u, we already have computed the sets N'_v and N_v^{ext} for each hit child v of u in the modular decomposition tree T. The case when u is a (hit) leaf is trivial: both functions return as neighbourhood of x the vertex of G that labels this leaf. Now consider an internal node u, we show how to determine the type of H_u and an optimal (resp. external-optimal) representation.

For lack of space, we do not describe the (easier) cases where node u is *parallel* or *series* , but we sketch the case where u is *prime*. In this case, P_u admits a unique permutation model (π_1, π_2) (up to symmetries) and therefore all completions H_u of $P_u + x$ respect (π_1, π_2). Then, we can get a minimal completion H_u by using the algorithm of Theorem 1, in $O(k)$ time, where k is the number of children of node u in T. We have to pay attention to the fact that there may have other insertion positions in (π_1, π_2) defining the same completion H_u, some of them possibly resulting in a smaller type for H_u, in which case we must use one of smallest type in order to get an optimal representation for H_u. Fortunately, Theorem 4 of [3] implies that there exists at most one other insertion position and that it can be obtained from the first one in $O(k)$ time. Consequently, our algorithm simply computes the two possible insertion positions. For each of them, it checks the two conditions $N_\alpha^{ext} \subsetneq V[v_\alpha]$ and $N'_\alpha \subsetneq V[v_\alpha]$ for the vertices $v_\alpha \in V_u$ that have one endpoint next to one endpoint of x in π_1 or in π_2. Then, it decides which one of the two insertion positions gives the smaller type and uses it in an optimal representation $\mathcal{R}$ of H_u. Finally, we compute the set N'_u of neighbours of x in the completion resulting from $\mathcal{R}$ as in Definition 8. We do not detail the computation of the set N_u^{ext}, which is similar and much simpler (in particular it does not need the algorithm of Theorem 1).

With a careful implementation of the algorithm , we compute in $O(n)$ time the neighbourhood N' of x in a minimal completion H of $G + x$. Using the algorithm of [3], we transform the modular decomposition of G into the one of H in time $O(n)$. By incrementally applying this algorithm we conclude:

Theorem 3 *There is an $O(n^2)$ time algorithm computing a minimal permutation completion of an arbitrary input graph.*

References

[1] A. Bergeron, C. Chauve, F. de Montgolfier, and M. Raffinot. Computing common intervals of k permutations, with applications to modular decomposition of graphs. In *ESA*, number 3669 in LNCS, pages 779–790, 2005.

[2] Pablo Burzyn, Flavia Bonomo, and Guillermo Durán. NP-completeness results for edge modification problems. *Discrete Applied Mathematics*, 154(13):1824–1844, 2006.

[3] Christophe Crespelle and Christophe Paul. Fully dynamic algorithm for recognition and modular decomposition of permutation graphs. *Algorithmica*, 58(2):405–432, 2010.

[4] Christophe Crespelle and Ioan Todinca. An $O(n^2)$-time algorithm for the minimal interval completion problem. *Theor. Comput. Sci.*, 494:75–85, 2013.

[5] P. Heggernes. Minimal triangulations of graphs: A survey. *Discrete Math.*, 306(3):297–317, 2006.

[6] P. Heggernes, F. Mancini, and C. Papadopoulos. Minimal comparability completions of arbitrary graphs. *Discrete Applied Mathematics*, 156(5):705–718, 2008.

[7] P. Heggernes, J. A. Telle, and Y. Villanger. Computing minimal triangulations in time $O(n^{\alpha \log n}) = o(n^{2.376})$. *SIAM J. Discrete Math.*, 19(4):900–913, 2005.

[8] Pinar Heggernes and Federico Mancini. Minimal split completions. *Discrete Applied Mathematics*, 157(12):2659–2669, 2009.

[9] Daniel Lokshtanov, Federico Mancini, and Charis Papadopoulos. Characterizing and computing minimal cograph completions. *Discrete Applied Mathematics*, 158(7):755–764, 2010.

[10] Federico Mancini. *Graph Modification Problems Related to Graph Classes*. PhD thesis, University of Bergen, Norway, 2008.

[11] T. Ohtsuki, H. Mori, T. Kashiwabara, and T. Fujisawa. On minimal augmentation of a graph to obtain an interval graph. *Journal of Computer and System Sciences*, 22(1):60–97, 1981.

[12] Tatsuo Ohtsuki. A fast algorithm for finding an optimal ordering for vertex elimination on a graph. *SIAM J. Comput.*, 5(1):133–145, 1976.

[13] I. Rapaport, K. Suchan, and I. Todinca. Minimal proper interval completions. *Information Processing Letters*, 5:195–202, 2008.

[14] Donald J. Rose, Robert Endre Tarjan, and George S. Lueker. Algorithmic aspects of vertex elimination on graphs. *SIAM J. Comput.*, 5(2):266–283, 1976.

[15] Mihalis Yannakakis. Computing the minimum fill-in is NP-complete. *SIAM. J. on Algebraic and Discrete Methods*, 2(1):77–79, 1981.