# ON THE RECOGNITION OF P4-INDIFFERENT GRAPHS

Romeo Rizzi

February 2001

Technical Report # DIT-02-051

Also : in Discrete Mathematics 239 (2001) 161--169.

# On the Recognition of $P_4$-Indifferent Graphs

Romeo Rizzi

**BRICS**[*]
Department of Computer Science
University of Aarhus
Ny Munkegade
DK-8000 Aarhus C, Denmark
e-mail: `romeo@cwi.nl`

**Abstract**

A simple graph is *$P_4$-indifferent* if it admits a total order $<$ on its nodes such that every chordless path with nodes $a, b, c, d$ and edges $ab, bc, cd$ has $a < b < c < d$ or $a > b > c > d$. $P_4$-indifferent graphs generalize indifferent graphs and are perfectly orderable. Recently, Hoàng, Maffray and Noy gave a characterization of $P_4$-indifferent graphs in terms of forbidden induced subgraphs. We clarify their proof and describe a linear time algorithm to recognize $P_4$-indifferent graphs. When the input is a $P_4$-indifferent graph, then the algorithm computes an order $<$ as above.

**Key words**: $P_4$-indifference, linear time, recognition, modular decomposition.

## 1 Introduction

A simple graph $G = (V, E)$ is called *perfectly orderable* if there exists a total order $<$ on $V$ with the following property: if $a, b, c, d \in V$ induce a chordless path (in jargon, a $P_4$) with edges $ab, bc$ and $cd$, then, either $a > b$, or $d > c$. The interest in perfectly orderable graphs is motivated by the notable fact, pointed out by Chvátal [3], that the greedy coloring algorithm applied along the order always produces an optimal coloring. A simple graph $G = (V, E)$ is called *$P_4$-indifferent* if it admits a *$P_4$-indifferent order*, that is, a total order $<$ on $V$ with the following property: if $a, b, c, d \in V$ induce a $P_4$ with edges $ab, bc$ and $cd$, then, either $a < b < c < d$, or $a > b > c > d$. The $P_4$-indifferent graphs were introduced in [8] as a polynomially recognizable subclass of perfectly orderable graphs. The interest in the subclass of $P_4$-indifferent graphs comes from the fact that the recognition of perfectly orderable graphs in general is *NP*-complete [10]. Recently, Hoàng, Maffray and Noy [7] gave a characterization of $P_4$-indifferent graphs in terms of forbidden induced subgraphs. We clarify their proof and give a linear time algorithm to recognize $P_4$-indifferent graphs. When the input of the algorithm is a $P_4$-indifferent graph, then a $P_4$-indifferent order is also obtained. Our algorithm is based on the modular decomposition of the input graph.

After having completed the present work, we came to know that a linear time recognition algorithm had been recently obtained by Habib, Paul and Viennot in [6]. A main original contribution of this paper is however a slight simplification in the proof of the result of Hoàng, Maffray and Noy [7] with a more clear understanding of the properties and the relationships among certain subclasses of interval graphs.

[*]Basic Research in Computer Science,
Centre of the Danish National Research Foundation.

As usual, $C_k$ denotes the chordless cycle on $k$ vertices. If $S \subset V$, then $G[S]$ denotes the subgraph of $G$ induced by $S$, i.e. $G[S] = (S, \{uv \in E : u, v \in S\})$. When we say "$G$ contains (a graph) $H$", we mean "$G$ contains $H$ as induced subgraph". Note that, if $G$ is $P_4$-indifferent, then every induced subgraph of $G$ is $P_4$-indifferent. The starting point and main inspiration of the present work is the following forbidden induced subgraph characterization of $P_4$-indifferent graphs, due to Hoàng, Maffray and Noy [7].

**Theorem 1.1** *A graph is a $P_4$-indifferent graph if and only if it contains no $C_k$ with $k \geq 5$ and none of the graphs $F_1, \ldots, F_8$ shown in Fig. 1.*

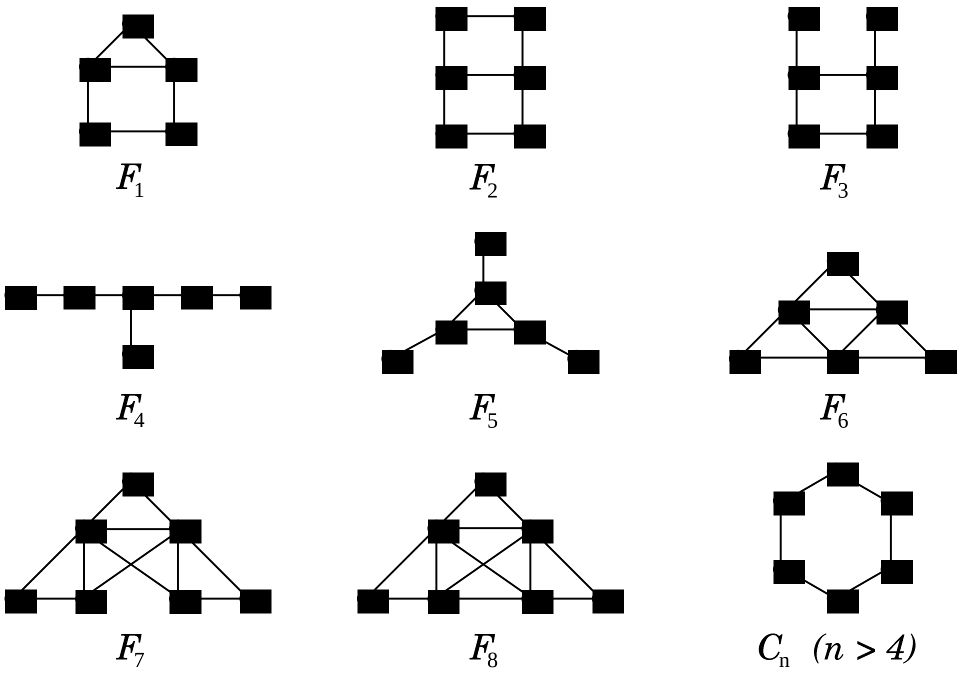

Figure 1: Forbidden subgraphs for $P_4$-indifferent graphs.

## 2 Interval graphs which are $P_4$-indifferent

An *interval graph* is any simple graph which admits an interval representation.

**Definition 2.1 (interval representation)** *Let $G = (V, E)$ be a simple graph with $n$ nodes. Two integers $l_v$ and $r_v$ with $l_v < r_v$ are associated to every node $v$ of $G$ so that $\{l_v : v \in V\} \cup \{r_v : v \in V\} = \{1, \ldots, 2n\}$. The following property is the main requirement: $uv \in E$ if and only if $l_u < l_v < r_u$ or $l_v < l_u < r_v$.*

In this section, we give a linear time algorithm, which, given an interval graph $G$, returns either an $F_4$ or an $F_7$ contained in $G$, or a $P_4$-indifferent order of $V$. A consequence is the following fact, already implicit in [7].

**Fact 2.2** *An interval graph is $P_4$-indifferent if and only if it contains no $F_4$ and no $F_7$.*

*Proof:* Is easy to check that neither $F_4$ nor $F_7$ are $P_4$-indifferent. If an interval graph $G$ with no $F_4$ and no $F_7$ is given as input to the algorithm, then a $P_4$-indifferent order is returned; hence $G$ is $P_4$-indifferent. □

Linear time algorithms to recognize interval graphs and compute interval representations of interval graphs are known [1, 4] (see p. 51 in [2] for an overview). Moreover, the following is a well-known [9] characterization of interval graphs in terms of excluded induced subgraphs.

**Lemma 2.3 (Lekkerkerker and Boland [9])** *A simple graph is an interval graph if and only if it contains none of the graphs shown in Fig. 2.*

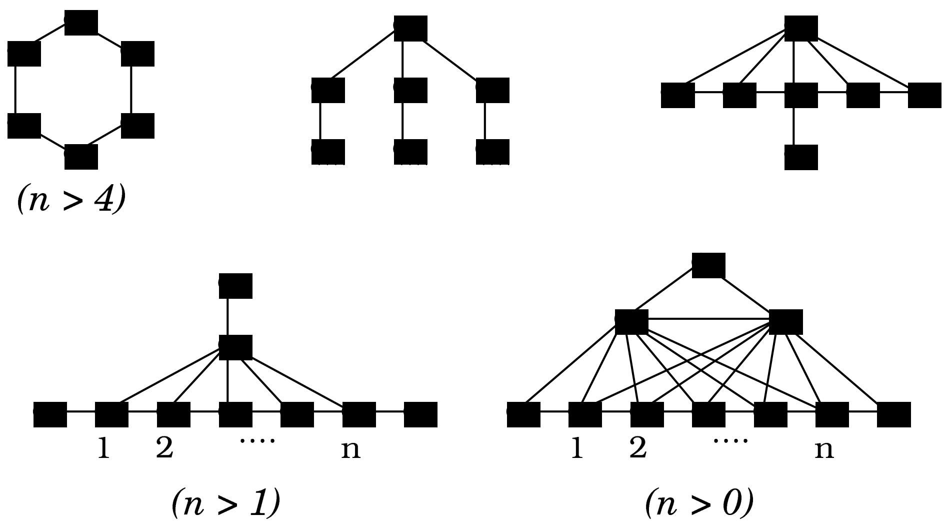

Figure 2: Forbidden subgraphs for interval graphs.

Our algorithm works on the interval representation of the input interval graph $G = (V, E)$. The algorithm scans the integers in the interval $[1, 2n]$ from left to right. During the scan, three sets of nodes $S_0, S_1$ and $S_2$ are maintained. For every node $v$, and for $j = 0, 1, 2$, let $t_v^j$ be the first instant in the interval $[1, 2n]$ for which $v \in S_j$. (Let $t_v^j = +\infty$ if $v$ never enters $S_j$). At every instant $i$, the sets $S_0, S_1$ and $S_2$ are as follows:

$S_0(i)$ contains a node $v \in V$ iif $l_v \leq i < r_v$;

$S_1(i)$ contains a node $v \in S_0(i)$ iif there exists a node $u$ with $l_v < r_u \leq i$;

$S_2(i)$ contains a node $v \in S_0(i)$ iif there exists a node $u$ with $l_v < r_u \leq i$ and $t_u^1 < l_v$.

In practice, a node $v$ is in $S_0(i)$ for $i \in [l_v, r_v)$. A node $v$, which ever enters $S_1$, will be in $S_1(i)$ for $i \in [t_v^1, r_v)$. A node $v$, which ever enters $S_2$, will be in $S_2(i)$ for $i \in [t_v^2, r_v)$.

When $i = r_v$, then we declare $v$ to be a *$u$-dangerous node* for all those nodes $u \in S_2(r_v)$ and such that $t_u^2 < l_v$.

This is the first phase of our algorithm. Note that, by reversing an interval representation of $G$, a second interval representation of $G$ is obtained. The second phase of our algorithm is identical to the first, only that it is performed on the reversed interval representation.

**Claim 2.4** *Assume a node $v$ to be declared $u$-dangerous both in the forward phase and in the backward phase. Then $G$ contains an $F_4$.*

*Proof:* It suffices to show that if $v$ is $u$-dangerous in the forward phase of the algorithm, then $r_v < r_u$ and there exists two nodes $a$ and $b$ such that $l_b < r_a < l_u < r_b < l_v$.

If $v$ is $u$-dangerous w.r.t. the forward phase, then $u \in S_2(r_v)$ (which accounts for $r_v < r_u$) and $t_u^2 < l_v$. Therefore, there exists a node $b$ with $r_b = t_u^2$ and $t_b^1 < l_u$. Finally, there exists a node $a$ with $l_b < r_a = t_b^1$. Obviously, $t_u^2 > l_u$. Summarizing, $l_b < r_a < l_u < r_b < l_v$. $\square$

The following relation $<^*$ on $V$ is equivalent to the one introduced in [7] after Remark 2.

- **Overlap rule.** If $u, v \in V$ with $l_u < l_v < r_u < r_v$, then $u <^* v$.
- **Containment rule.** If $v$ is declared $u$-dangerous in the forward phase, then $u <^* v$. If $v$ is declared $u$-dangerous in the backward phase, then $v <^* u$.

Note that $u <^* v$ implies $uv \in E$. Moreover, by Claim 2.4, when $G$ contains no $F_4$, then $<^*$ is antisymmetric.

**Claim 2.5** *If $G$ contains no $F_7$ and $<^*$ is antisymmetric, then the relation $<^*$ is acyclic.*

*Proof:* The following relation is clearly acyclic: $u <' v$ iff $l_u < l_v$. Therefore, in every cycle of $<^*$, a $v <^* u$ for which $v$ is $u$-dangerous (backwards), must appear. Let $z$ be the predecessor of $v$ in the cycle. We assume that $z \not<^* u$ since otherwise, considering $z <^* u$ instead of $z <^* v$ and $v <^* u$, a shorter cycle of $<^*$ is obtained. Let $a$ and $b$ be two nodes which cause $v$ to be $u$-dangerous, i.e., $r_v < l_b < r_u < l_a < r_b$.

*Case 1:* assume $r_z < l_b$. Since $vz \in E$, then $r_z > l_v$; hence $r_z > l_u$. If $l_z < l_u$ then $z <^* u$ by the overlap rule. Otherwise, if $l_z > l_u$ then $z$ is $u$-dangerous as well as $v$. Again $z <^* u$.

*Case 2:* assume $l_b < r_z < r_u$. If $l_z < l_u$ then $z <^* u$ by the overlap rule. Assume therefore $l_z > l_u$. Since $z <^* v$ and $r_z > r_v$, then the interval $[l_z, r_z]$ contains the interval $[l_v, r_v]$ and $v$ is $z$-dangerous (forwards). However $r_v < l_b < r_z < l_a$. Therefore $v$ is $z$-dangerous also in the backward phase, contrary to our assumptions.

*Case 3:* assume $r_z > r_u$. Since $z <^* v$ and $r_z > r_v$, then $v$ must be $z$-dangerous (forwards). If $r_z < l_a$, then $v$ is $z$-dangerous also in the backward phase, contrary to our assumptions. Assume therefore $r_z > l_a$. Let $b'$ and $a'$ be two nodes which cause $v$ to be $z$-dangerous (forwards), i.e., $l_{b'} < r_{a'} < l_z < r_{b'} < l_v$. If $r_{b'} < l_u$, then also $u$ is $z$-dangerous in the forward phase and by the containment rule $z <^* u$. Assume therefore $r_{b'} > l_u$. If $r_{a'} < l_u$, and since $r_{b'} < l_v$, then $v$ is $u$-dangerous also in the forward phase, contrary to our assumptions. Assume therefore $r_{a'} > l_u$. But now, $u, v, z, a, b, a', b'$ induce an $F_7$, contrary to our assumptions. □

By Claims 2.4 and 2.5, when $G$ contains no $F_4$ and no $F_7$, then there exists a total order $<^+$ on $V$ containing $<^*$.

**Claim 2.6** *If $<^*$ is antisymmetric and acyclic, then $<^+$ is a $P_4$-indifferent order.*

*Proof:* Let $a, b, c$ and $d$ be four nodes inducing a chordless path with edges $ab, bc$ and $cd$. By eventually exchanging $b$ with $c$ and $a$ with $d$, we can always assume that $l_b < l_c$. Hence, $l_b < l_c < r_b < r_c$, for otherwise $d$ could not be adjacent to $c$ without being adjacent to $a$. Therefore $l_b < r_a < l_c < r_b < l_d < r_c$ and $b <^+ c$.

If $r_c < r_d$ then $c <^+ d$. Otherwise, if $r_d < r_c$, then $d$ is $c$-dangerous in the forward phase and $c <^+ d$ anyhow.

If $r_a < r_b$ then $a <^+ b$. Otherwise, if $r_b < r_a$, then $a$ is $b$-dangerous in the backward phase and $a <^+ b$ anyhow. □

## 2.1 Running time and general outline of the algorithm

The forward phase (and hence the backward phase) of the algorithm is easily implemented to run in linear time. More precisely, the total cost of updating $S_0$ during one scan is $O(V)$, whereas the total cost of updating $S_1$ and $S_2$ is $O(E)$. Indeed, at every step in the interval $[1, 2n]$ a single node $v$ enters or leaves $S_0$. In case $v$ leaves $S_0$, then some neighbors of $v$ can enter $S_1$ or $S_2$ or be declared $v$-dangerous. For the sake of clarity, a formal description of the updates to be performed at *Step i* is given here below.

**Step ($i$):** Let $v$ be the node such that $i \in \{l_v, r_v\}$. If $i = l_v$, then node $v$ enters $S_0$. Otherwise, if $i = r_v$ then

(1) node $v$ exits $S_0$, $S_1$ and $S_2$;
(2) put in $S_1$ every node $u \in S_0 \setminus S_1$ and set $t_u^1 := i$;
(3) put in $S_2$ every node $u \in S_0 \setminus S_2$ with $t_v^1 < l_u$ and set $t_u^2 := i$;
(4) for every node $u \in S_2$ with $t_u^2 < l_v$, declare $v$ to be $u$-dangerous.

Note that only the nodes in $S_0$ can go into $S_1$ or $S_2$ or become $v$-dangerous. Moreover, all nodes in $S_0$ are neighbors of $v$.

After the two scans, if a node $v$ turns out to be $u$-dangerous both in the forward and in the backward phase, then the proof of Claim 2.4 shows how to produce an $F_4$ contained in $G$ in constant time. Assume therefore $<^*$ to be antisymmetric. Testing the acyclicity of $<^*$ amounts to test the acyclicity of a digraph with $V$ as vertex-set and with at most $|E|$ arcs. (Remember that $u <^* v$ implies $uv \in E$). It is well known that this can be done in linear time, while at the same time computing a total order $<^+$ on $V$ which contains $<^*$. (Every acyclic digraph contains a source. Keep removing source nodes one after the other. If all nodes get removed, then let $<^+$ be the order in which the nodes have been removed. Otherwise, if at a certain point no node is source, then a cycle is obtained in at most $n$ steps, going backwards starting from any node. Moreover, a chordless cycle can be easily obtained in linear time). If a chordless cycle $C$ is returned, then the proof of Claim 2.5 shows that the nodes in $C$ induce an $F_7$ in $G$. If the antisymmetric relation $<^*$ is acyclic, then the total order $<^+$ is $P_4$-indifferent by Claim 2.6.

## 3 Modules

If $u$ is adjacent to $v$ in a graph $G$, we say that *$u$ sees $v$* in $G$, otherwise we say that *$u$ misses $v$* in $G$. A *module* of an undirected simple graph $G = (V, E)$ is a non-empty set $X$ of nodes such that every node $v \in V \setminus X$ either sees all nodes in $X$ or no node in $X$. By definition, all singletons and $V$ itself are modules — called the *trivial* modules of $G$. A graph is *prime* if it has no nontrivial modules.

In Subsection 3.1, we describe a linear time algorithm to decide if a given prime graph is $P_4$-indifferent. In Subsection 3.2, we report some basic facts in modular decomposition theory and show how to reduce the recognition of $P_4$-indifferent graphs to the special case when the input graph is prime.

### 3.1 Prime graphs

In this subsection, we show that every prime graph is an interval graph, provided it contains no $C_k$ with $k \geq 5$ and none of the graphs $F_1, \ldots, F_8$ shown in Fig. 1. This result was first given in [7], while the key Lemma 3.1 already appeared in [8, 12]. Combining this with the algorithm in Section 2, we obtain a linear time algorithm to decide if a given prime graph is $P_4$-indifferent.

**Lemma 3.1 ([8, 12])** *If $G$ is a prime graph containing no $F_1, F_2, F_3$, then $G$ contains no $C_4$.*

*Proof:* Short proofs can be found in [7], [8] or [12]. □

**Corollary 3.2 ([7])** *Let $G$ be a prime graph containing no $C_k$ with $k \geq 5$ and none of the graphs $F_1, \ldots, F_8$. Then $G$ is an interval graph.*

*Proof:* By Lemma 3.1, $G$ contains no $C_4$. Check that each one of the forbidden induced subgraphs for interval graphs, given in Fig. 2, contains a $C_k$ ($k \geq 4$) or one of $F_4, \ldots, F_8$. □

Let $G$ be the prime graph given as input. Thanks to the algorithm of Booth and Lueker [1], we can decide in linear time if $G$ is an interval graph. If $G$ is not an interval graph, then the algorithm of Booth and Lueker returns (in linear time) one of the graphs shown in Fig. 2. Hence, by Corollary 3.2, we can produce in linear time a $C_k$ with $k \geq 5$ or one of $F_1, \ldots, F_8$. Note that none of these graphs is $P_4$-indifferent. Therefore, $G$ is not $P_4$-indifferent.

If $G$ is an interval graph, then the algorithm of Booth and Lueker returns (in linear time) an interval representation of $G$. Now we apply the algorithm given in Section 2. This linear time algorithm will (1) either return an $F_4$ or an $F_7$ contained in $G$, hence proving that $G$ is not $P_4$-indifferent; (2) or return a $P_4$-indifferent order for $G$.

### 3.2 Modular decomposition

In this subsection, we show how to reduce the recognition of $P_4$-indifferent graphs to the special case when the input graph is prime. The reduction is based on the notion of modular decomposition of an undirected graph as introduced by Gallai in [5]. This decomposition is also known as *substitution decomposition*, *prime tree decomposition*, and *$X$-join decomposition.* We refer to [2, 13] for an introduction to modular decompositions and to [11] for a survey on the many aspects of this subject. The few properties needed are given 'de facto' in Definition 3.4 here below. The existence of a linear time algorithm to compute the modular decomposition of the input graph $G$ is fundamental to our solution. In 1994, McConnell and Spinrad [14, 13] gave a linear time algorithm to compute the modular decomposition of any graph. We will not go into the details of their algorithm either, and assume the modular decomposition of $G$ to exist and to be given as part of the input.

The following observation points out the role of modules in recognizing $P_4$-indifferent graphs and in computing $P_4$-indifferent orders.

**Observation 3.3** *Let $X$ be a module of $G$ and let $x$ be any node in $X$. If $x_1, \ldots, x_p$ is a $P_4$-indifferent order w.r.t. $G[X]$ and $u_1, \ldots, u_i = x, \ldots, u_q$ is a $P_4$-indifferent order w.r.t. $G[V \setminus X \cup \{x\}]$, then $u_1, \ldots, u_{i-1}, x_1, \ldots, x_p, u_{i+1}, \ldots, u_q$ is $P_4$-indifferent w.r.t. $G$.*

*Proof:* If $X$ is a module of $G$, then every $P_4$ of $G$ has either zero, or one, or four nodes in $X$, and if it has one, then this node is a leaf of the $P_4$. Clearly, every $P_4$ that has zero or four nodes in $X$ is properly ordered. Moreover, if $ab, bc, cd$ is a $P_4$ of $G$ with solely $d$ in $X$, then $ab, bc, cd$ is properly ordered as well as $ab, bc, cx$. $\square$

**Definition 3.4 (modular decomposition of an undirected graph $G = (V, E)$)** *An out-directed tree $T$ with root $r$ is given. The leaves of $T$ correspond to the nodes in $V$. Every non-leaf node has at least two children and is given a label in $\{0, 1, 2\}$. For every node $t$ of $T$, let $V_t$ be the set of those nodes in $V$ which correspond to the leaves which can be reached from $t$ in $T$. Let $t_1, \ldots, t_k$ be the children of $t$. Let $\hat{V}_t$ be any subset of $V_t$ such that $|\hat{V}_t \cap V_{t_1}| = \ldots = |\hat{V}_t \cap V_{t_k}| = 1$. We require the following properties to hold:*

- *$V_t$ is a module of $G$ for every node $t$ of $T$;*
- *if $t$ is labeled 2, then $G[\hat{V}_t]$ is prime;*
- *if $t$ is labeled 1, then $G[\hat{V}_t]$ is a complete graph;*
- *if $t$ is labeled 0, then $\overline{G}[\hat{V}_t]$ is a complete graph.*

Computing a $P_4$-indifferent order for $G$ corresponds to compute a $P_4$-indifferent order for $G[V_r]$. By Observation 3.3, and by the properties expressed in Definition 3.4, this can be done recursively as follows. Let $t$ be any node of $T$. Let $t'_1, \ldots, t'_k$ be a $P_4$-indifferent order for $G[\hat{V}_t]$. For $i = 1, \ldots k$, let $t_i$ be the child of $t$ such that $t'_i \in V_{t_i}$ and let $u^i_1, \ldots, u^i_{h_i}$ be a $P_4$-indifferent order for $G[V_{t_i}]$. Then a $P_4$-indifferent order for $G[V_t]$ is obtained by juxtaposing the $P_4$-indifferent orders for $G[V_{t_1}], \ldots, G[V_{t_k}]$ as follows: $u_1^1, \ldots, u_{h_1}^1, \ldots, u_1^k, \ldots, u_{h_k}^k$. It only remains to show how to compute a $P_4$-indifferent order for $G[\hat{V}_t]$ or find a forbidden subgraph in $G[\hat{V}_t]$ in linear time. If $t$ is labeled 0, then $\overline{G}[\hat{V}_t]$ is a complete graph and any total order on $\hat{V}_t$ is $P_4$-indifferent. The same conclusion holds if $t$ is labeled 1 and $G[\hat{V}_t]$ is a complete graph. When $t$ is labeled 2, then $G[\hat{V}_t]$ is prime and we can resort on the linear time algorithm given in Subsection 3.1.

## 4 Acknowledgments

Thanks are due to the anonymous referee. His advice helped improving the presentation. I also thank Frédéric Maffray for his careful reading and Christophe Paul, Michel Habib, and Laurent Viennot, for sending me a copy of their manuscript [6].

## References

[1] K.S. Booth and G.S. Lueker, Linear algorithms to recognize interval graphs and test for the consecutive ones property. Seventh Annual ACM Symposium on Theory of Computing (Albuquerque, N. M., 1975), pp. 255–265. Assoc. Comput. Mach., New York, 1975.

[2] A. Brandstädt, V.B. Le and J.P. Spinrad Graph classes: a survey. *SIAM Monographs on Discrete Mathematics and Applications.* (SIAM), Philadelphia, PA, 1999.

[3] V. Chvàtal, Perfectly ordered graphs. *Topics on perfect graphs*, 63–65, North-Holland Math. Stud., 88, North-Holland, Amsterdam-New York, 1984.

[4] D.G. Corneil, S. Olariu and L. Stewart, The ultimate interval graph recognition algorithm? *Proceedings of the Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, 175–180, San Francisco, 1998.

[5] T. Gallai, Transitiv orientierbare Graphen. (German) *Acta Math. Acad. Sci. Hungar* 18 1967 25–66.

[6] M. Habib, C. Paul and L. Viennot, Linear time recognition of $P_4$-indifference graphs. *manuscript.*

[7] C.T. Hoàng, F. Maffray and M. Noy, A characterization of $P_4$-indifference graphs. *J. Graph Theory* 31 (1999), no. 3, 155–162.

[8] C.T. Hoàng and B.A. Reed, Some classes of perfectly orderable graphs. *J. Graph Theory* 13 (1989), no. 4, 445–463.

[9] C.G. Lekkerkerker and J.Ch. Boland, Representation of a finite graph by a set of intervals on the real line. *Fund. Math.* 51 1962/1963 45–64.

[10] M. Middendorf and F. Pfeiffer, On the complexity of recognizing perfectly orderable graphs. *Discrete Math.* 80 (1990), no. 3, 327–333.

[11] R.H. Mőhring, Algorithmic aspects of comparability graphs and interval graphs. Graphs and order (Banff, Alta., 1984), 41–101. Also in: NATO Adv. Sci. Inst. Ser. C: Math. Phys. Sci., 147, Reidel, Dordrecht-Boston, Mass., 1985.

[12] S. Olariu, Weak bipolarizable graphs. *Discrete Math.* 74 (1989), no. 1-2, 159–171.

[13] R.M. McConnell and J.P. Spinrad, Modular decomposition and transitive orientation. *Discrete Math.* 201 (1999), no. 1-3, 189–241.

[14] R.M. McConnell and J.P. Spinrad, Linear-time modular decomposition and efficient transitive orientation of comparability graphs. *Proceedings of the Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, 536–545, New York, 1994.