RWTH Aachen

Department of Computer Science
Technical Report

The Longest Path Problem is Polynomial on Interval Graphs

Kyriaki Ioannidou
George B. Mertzios
Stavros D. Nikolopoulos

ISSN 0935–3232 · Aachener Informatik Berichte · AIB-2009-11

RWTH Aachen · Department of Computer Science · April 2009

The publications of the Department of Computer Science of *RWTH Aachen University* are in general accessible through the World Wide Web.

http://aib.informatik.rwth-aachen.de/

The Longest Path Problem is Polynomial on Interval Graphs

Kyriaki Ioannidou[1]*, George B. Mertzios[2], and Stavros D. Nikolopoulos[1]*

[1] Department of Computer Science, University of Ioannina, Greece
{kioannid, stavros}@cs.uoi.gr
[2] Department of Computer Science, RWTH Aachen, Germany
mertzios@cs.rwth-aachen.de

Abstract. The longest path problem is the problem of finding a path of maximum length in a graph. Polynomial solutions for this problem are known only for small classes of graphs, while it is NP-hard on general graphs, as it is a generalization of the Hamiltonian path problem. Motivated by the work of Uehara and Uno in [22], where they left the longest path problem open for the class of interval graphs, in this paper we show that the problem can be solved in polynomial time on interval graphs. The proposed algorithm runs in $O(n^4)$ time, where n is the number of vertices of the input graph, and bases on a dynamic programming approach.

Keywords: Longest path problem, interval graphs, polynomial algorithm, complexity, dynamic programming.

1 Introduction

A well studied problem in graph theory with numerus applications is the Hamiltonian path problem, i.e., the problem of determining whether a graph is Hamiltonian; a graph is said to be Hamiltonian if it contains a Hamiltonian path, that is, a simple path in which every vertex of the graph appears exactly once. Even if a graph is not Hamiltonian, it makes sense in several applications to search for a longest path, or equivalently, to find a maximum induced subgraph of the graph which is Hamiltonian. However, finding a longest path seems to be more difficult than deciding whether or not a graph admits a Hamiltonian path. Indeed, it has been proved that even if a graph has a Hamiltonian path, the problem of finding a path of length $n - n^{\varepsilon}$ for any $\varepsilon < 1$ is NP-hard, where n is the number of vertices of the graph [16]. Moreover, there is no polynomial-time constant-factor approximation algorithm for the longest path problem unless P=NP [16]. For related results see also [3, 8–10, 24, 25].

It is clear that the longest path problem is NP-hard on every class of graphs, on which the Hamiltonian path problem is NP-complete. The Hamiltonian path problem is known to be NP-complete in general graphs [11, 12], and remains NP-complete even when restricted to some small classes of perfect graphs, such as split graphs [14], chordal bipartite graphs, split strongly chordal graphs [19], circle graphs [6], planar graphs [12], and grid graphs [15]. However, it makes sense to investigate the tractability of the longest path problem on the classes for which the Hamiltonian path problem admits polynomial time algorithms. Such classes include interval graphs [18], circular-arc graphs [7], convex bipartite graphs [19], co-comparability graphs [5]. Note that the case of proper interval graphs is easy, since all connected proper interval graphs have a Hamiltonian path [2].

* This research is co-financed by E.U.-European Social Fund (80%) and the Greek Ministry of Development-GSRT (20%).

In contrast to the Hamiltonian path problem, there are few known polynomial time solutions for the longest path problem, and these restrict to trees and some small graph classes. Specifically, a linear time algorithm for finding a longest path in a tree was proposed by Dijkstra around 1960, a formal proof of which can be found in [4]. Later, through a generalization of Dijkstra's algorithm for trees, Uehara and Uno [22] solved the longest path problem for weighted trees and block graphs in linear time and space, and for cacti in $O(n^2)$ time and space, where n and m denote the number of vertices and edges of the input graph, respectively. More recently, polynomial algorithms have been proposed that solve the longest path problem on bipartite permutation graphs in $O(n)$ time and space [23], and on ptolemaic graphs in $O(n^5)$ time and $O(n^2)$ space [21].

Furthermore, Uehara and Uno in [22] introduced a subclass of interval graphs, namely interval biconvex graphs, which is a superclass of proper interval and threshold graphs, and solved the longest path problem on this class in $O(n^3(m + n \log n))$ time. As a corollary, they showed that a longest path of a threshold graph can be found in $O(n + m)$ time and space. They left open the complexity of the longest path problem on interval graphs.

In this paper, we resolve the open problem posed in [22] by showing that the longest path problem admits a polynomial time solution on interval graphs. Interval graphs form an important and well-known class of perfect graphs [14]; a graph G is an interval graph if its vertices can be put in a one-to-one correspondence with a family of intervals on the real line, such that two vertices are adjacent in G if and only if their corresponding intervals intersect. In particular, we propose an algorithm for solving the longest path problem on interval graphs which runs in $O(n^4)$ time using a dynamic programming approach. Thus, not only we answer the question left open by Uehara and Uno in [22], but also improve the known time complexity of the problem on interval biconvex graphs, a subclass of interval graphs [22].

Interval graphs form a well-studied class of perfect graphs, have important properties, and admit polynomial time solutions for several problems that are NP-complete on general graphs (see e.g. [1, 14, 17]). Moreover, interval graphs have received a lot of attention due to their applicability to DNA physical mapping problems [13], and find many applications in several fields and disciplines such as genetics, molecular biology, scheduling, VLSI circuit design, archaeology and psychology [14].

The rest of this paper is organized as follows. In Section 2, we review some properties of interval graphs and introduce the notion of normal paths, which is central for our algorithm. In Section 3, we present our algorithm for solving the longest path problem on an interval graph, which includes three phases. In Section 4 we prove the correctness and compute the time complexity of our algorithm. Finally, some concluding remarks are given in Section 5.

2 Theoretical Framework

We consider finite undirected graphs with no loops or multiple edges. For a graph G, we denote its vertex and edge set by $V(G)$ and $E(G)$, respectively. An edge is a pair of distinct vertices $u, v \in V(G)$, and is denoted by uv. Let S be a set of vertices of a graph G. Then, the cardinality of the set S is denoted by $|S|$ and the subgraph of G induced by S is denoted by $G[S]$. The set $N(v) = \{u \in V(G) : uv \in E(G)\}$ is called the *neighborhood* of the vertex $v \in V(G)$ in G, sometimes denoted by $N_G(v)$ for clarity reasons. The set $N[v] = N(v) \cup \{v\}$ is called the *closed neighborhood* of the vertex $v \in V(G)$.

Let G be a graph and let $P = (v_1, v_2, \ldots, v_{i-1}, v_i, v_{i+1}, \ldots, v_j, v_{j+1}, v_{j+2}, \ldots, v_k)$ and $P_0 = (v_i, v_{i+1}, \ldots, v_j)$ be two paths of the graph G. Sometimes, we shall denote the path P by $P = (v_1, v_2, \ldots, v_{i-1}, P_0, v_{j+1}, v_{j+2}, \ldots, v_k)$. Moreover, we denote by $V(P)$ the set of vertices in the path P, and define the *length* of the path P to be the number of vertices in P, i.e., $|P| = |V(P)|$. We call *right endpoint* of a path $P = (v_1, v_2, \ldots, v_k)$ the last vertex v_k of P.

2.1 Structural Properties of Interval Graphs

A graph G is an *interval graph* if its vertices can be put in a one-to-one correspondence with a family F of intervals on the real line such that two vertices are adjacent in G if and only if the corresponding intervals intersect; F is called an *intersection model* for G [1]. The class of interval graphs is *hereditary*, that is, every induced subgraph of an interval graph G is also an interval graph. Ramalingam and Rangan [20] proposed a numbering of the vertices of an interval graph; they stated the following lemma.

Lemma 1. *(Ramalingam and Rangan [20]): The vertices of any interval graph G can be numbered with integers $1, 2, \ldots, |V(G)|$ such that if $i < j < k$ and $ik \in E(G)$, then $jk \in E(G)$.*

As shown in [20], the proposed numbering, which results after sorting the intervals of the intersection model of a graph G on their right ends [1], can be obtained in $O(|V(G)| + |E(G)|)$ time. An ordering of the vertices according to this numbering is found to be quite useful in solving some graph-theoretic problems on interval graphs [1, 20]. Throughout the paper, such an ordering is called a *right-end ordering* of G. Let u and v be two vertices of G; if π is a right-end ordering of G, denote $u <_\pi v$ if u appears before v in π. In particular, if $\pi = (u_1, u_2, \ldots, u_{|V(G)|})$ is a right-end ordering of G, then $u_i <_\pi u_j$ if and only if $i < j$.

The following lemma appears to be useful in obtaining some important results.

Lemma 2. *Let G be an interval graph, and let π be a right-end ordering of G. Let $P = (v_1, v_2, \ldots, v_k)$ be a path of G, and let $v_\ell \notin V(P)$ be a vertex of G such that $v_1 <_\pi v_\ell <_\pi v_k$ and $v_\ell v_k \notin E(G)$. Then, there exist two consecutive vertices v_{i-1} and v_i in P, $2 \le i \le k$, such that $v_{i-1}v_\ell \in E(G)$ and $v_\ell <_\pi v_i$.*

Proof. Consider the intersection model F of G, from we each we obtain the right-end ordering π of G. Let I_i denote the interval which corresponds to the vertex v_i in F, and let $l(I_i)$ and $r(I_i)$ denote the left and the right endpoint of the interval I_i, respectively. Without loss of generality, we may assume that all values $l(I_i)$ and $r(I_i)$ are distinct. Since $P = (v_1, v_2, \ldots, v_k)$ is a path from v_1 to v_k, it is clear from the intersection model F of G that at least one vertex of P sees v_ℓ. Recall that $v_k v_\ell \notin E(G)$; let v_{i-1}, $2 \le i \le k$, be the last vertex of P such that $v_{i-1}v_\ell \in E(G)$, i.e., $v_j v_\ell \notin E(G)$ for every index j, $i \le j \le k$. Thus, since $v_\ell <_\pi v_k$, it follows that $r(I_\ell) < l(I_j) < r(I_j)$ for every index j, $i \le j \le k$ and, thus, $v_\ell <_\pi v_j$. Therefore, in particular, $v_\ell <_\pi v_i$. This completes the proof. $\square$

2.2 Normal Paths

Our algorithm for constructing a longest path of an interval graph G uses a specific type of paths, namely normal paths. We next define the notion of a normal path of an interval graph G.

Definition 1. *Let G be an interval graph, and let π be a right-end ordering of G. The path $P = (v_1, v_2, \ldots, v_k)$ of G is called a* normal path*, if v_1 is the leftmost vertex of $V(P)$ in π, and for every i, $2 \leq i \leq k$, the vertex v_i is the leftmost vertex of $N(v_{i-1}) \cap \{v_i, v_{i+1}, \ldots, v_k\}$ in π.*

The notion of a normal path of an interval graph G is a generalization of the notion of a typical path of G; the path $P = (v_1, v_2, \ldots, v_k)$ of an interval graph G is called a *typical* path, if v_1 is the leftmost vertex of $V(P)$ in π. The notion of a typical path was introduced by Arikati and Rangan [1], in order to solve the path cover problem on interval graphs; they proved the following result.

Lemma 3. *(Arikati and Rangan [1]): Let P be a path of an interval graph G. Then there exists a typical path P' in G such that $V(P') = V(P)$.*

The following lemma is the basis of our algorithm for solving the longest path problem on interval graphs.

Lemma 4. *Let P be a path of an interval graph G. Then there exists a normal path P' of G, such that $V(P') = V(P)$.*

Proof. Let G be an interval graph, let π be a right-end ordering of G, and let $P = (v_1, v_2, \ldots, v_k)$ be a path of G. If $k = 1$, the lemma clearly holds. Suppose that $k \geq 2$. We will prove that for every index i, $2 \leq i \leq k$, there exists a path $P_i = (v'_1, v'_2, \ldots, v'_k)$, such that $V(P_i) = V(P)$, v'_1 is the leftmost vertex of $V(P_i)$ in π, and for every index j, $2 \leq j \leq i$, the vertex v'_j is the leftmost vertex of $N(v'_{j-1}) \cap \{v'_j, v'_{j+1}, \ldots, v'_k\}$ in π. The proof will be done by induction on i.

Due to Lemma 3, we may assume that $P = (v_1, v_2, \ldots, v_k)$ is typical, i.e., that v_1 is the leftmost vertex of $V(P)$ in π. Let $i = 2$. Assume that $v_j \in V(P)$, $j > 2$, is the leftmost vertex of $N(v_1) \cap \{v_2, v_3, \ldots, v_k\}$ in π. Then, since $G[V(P)]$ is an interval graph, $v_1 <_\pi v_j <_\pi v_2$, and $v_1v_2, v_1v_j \in E(G)$, it follows that $N[v_j] \cap \{v_1, v_2, \ldots, v_k\} \subseteq N[v_2] \cap \{v_1, v_2, \ldots, v_k\}$. Thus, there exists a path

$$P_2 = (v'_1, v'_2, \ldots, v'_k) = (v_1, v_j, v_{j-1}, \ldots, v_3, v_2, v_{j+1}, v_{j+2} \ldots, v_k)$$

of G, such that $V(P_2) = V(P)$, v'_1 is the leftmost vertex of $V(P_2)$ in π, and v'_2 is the leftmost vertex of $N(v'_1) \cap \{v'_2, v'_3, \ldots, v'_k\}$ in π. This proves the induction basis.

Consider now an arbitrary index i, $2 \leq i \leq k-1$, and let $P_i = (v'_1, v'_2, \ldots, v'_k)$ be a path of G, such that $V(P_i) = V(P)$, v'_1 is the leftmost vertex of $V(P_i)$ in π, and for every index j, $2 \leq j \leq i$, the vertex v'_j is the leftmost vertex of $N(v'_{j-1}) \cap \{v'_j, v'_{j+1}, \ldots, v'_k\}$ in π. In particular, it follows that the subpath $(v'_1, v'_2, \ldots, v'_i)$ of P_i is normal. We will now prove that for any vertex $v'_\ell \in \{v'_{i+1}, v'_{i+2}, \ldots, v'_k\}$, where $v'_\ell <_\pi v'_i$, it holds $v'_\ell v'_i \in E(G)$. Indeed, suppose otherwise that $v'_\ell v'_i \notin E(G)$, for such a vertex v'_ℓ. Then, since $v'_1 <_\pi v'_\ell <_\pi v'_i$, it follows by Lemma 2 that there are two consecutive vertices v'_{j-1} and v'_j in P_i, $2 \leq j \leq i$, such that $v'_{j-1}v'_\ell \in E(G)$ and

$v'_\ell <_\pi v'_j$. Thus, v'_j is not the leftmost vertex of $N(v'_{j-1}) \cap \{v'_j, v'_{j+1}, \ldots, v'_\ell, \ldots, v'_k\}$ in π, which is a contradiction. Therefore, for any vertex $v'_\ell \in \{v'_{i+1}, v'_{i+2}, \ldots, v'_k\}$, where $v'_\ell <_\pi v'_i$, it holds $v'_\ell v'_i \in E(G)$.

Assume that $v'_j \in V(P_i)$, $j > i+1$, is the leftmost vertex of $N(v'_i) \cap \{v'_{i+1}, v'_{i+2}, \ldots, v'_k\}$ in π. Consider first the case where $v'_i <_\pi v'_j$. Then, for every vertex $v'_\ell \in \{v'_{i+1}, v'_{i+2}, \ldots, v'_k\}$ it holds $v'_i <_\pi v'_\ell$. Indeed, suppose otherwise that $v'_\ell <_\pi v'_i <_\pi v'_j$ for such a vertex v'_ℓ. Then, as we have proved above, $v'_\ell v'_i \in E(G)$, which is a contradiction, since v'_j is the leftmost vertex of $N(v'_i) \cap \{v'_{i+1}, v'_{i+2}, \ldots, v'_k\}$ in π and $v'_\ell <_\pi v'_j$. Thus, $v'_i <_\pi v'_\ell$ for every vertex $v'_\ell \in \{v'_{i+1}, v'_{i+2}, \ldots, v'_k\}$. Therefore, since $G[V(P_i)]$ is an interval graph, $v'_i <_\pi v'_j <_\pi v'_{i+1}$, and $v'_i v'_{i+1}, v'_i v'_j \in E(G)$, it follows that $N[v'_j] \cap \{v'_i, v'_{i+1}, \ldots, v'_k\} \subseteq N[v'_{i+1}] \cap \{v'_i, v'_{i+1}, \ldots, v'_k\}$. Then, there exists the path

$$P_{i+1} = (v''_1, v''_2, \ldots, v''_i, v''_{i+1}, \ldots, v''_k) = (v'_1, v'_2, \ldots, v'_i, v'_j, v'_{j-1}, \ldots, v'_{i+2}, v'_{i+1}, v'_{j+1}, \ldots, v'_k)$$

of G, such that $V(P_{i+1}) = V(P_i)$, v''_1 is the leftmost vertex of $V(P_{i+1})$ in π, and for every index j, $2 \leq j \leq i+1$, the vertex v''_j is the leftmost vertex of $N(v''_{j-1}) \cap \{v''_j, v''_{j+1}, \ldots, v''_k\}$ in π.

Consider now the case where $v'_j <_\pi v'_i$. Then, v'_j is the leftmost vertex of $\{v'_{i+1}, v'_{i+2}, \ldots, v'_k\}$ in π. Indeed, suppose otherwise that $v'_\ell <_\pi v'_j <_\pi v'_i$ for a vertex $v'_\ell \in \{v'_{i+1}, v'_{i+2}, \ldots, v'_k\}$. Then, as we have proved above, $v'_\ell v'_i \in E(G)$, which is a contradiction, since v'_j is the leftmost vertex of $N(v'_i) \cap \{v'_{i+1}, v'_{i+2}, \ldots, v'_k\}$ in π and $v'_\ell <_\pi v'_j$. Thus, there exists by Lemma 3 a typical path P_0, such that $V(P_0) = \{v'_{i+1}, v'_{i+2}, \ldots, v'_k\}$. Since P_0 is typical and v'_j is the leftmost vertex of $V(P_0)$ in π, it follows that v'_j is the first vertex of P_0. Then, since $v'_i v'_j \in E(G)$, there exists the path

$$P_{i+1} = (v''_1, v''_2, \ldots, v''_i, v''_{i+1}, \ldots, v''_k) = (v'_1, v'_2, \ldots, v'_i, P_0)$$

of G, such that $V(P_{i+1}) = V(P_i)$, v''_1 is the leftmost vertex of $V(P_{i+1})$ in π, and for every index j, $2 \leq j \leq i+1$, the vertex v''_j is the leftmost vertex of $N(v''_{j-1}) \cap \{v''_j, v''_{j+1}, \ldots, v''_k\}$ in π. This proves the induction step.

Thus, the path $P' = P_k$ is a normal path of G, such that $V(P') = V(P)$. □

3 Interval Graphs and the Longest Path Problem

In this section we present our algorithm, which we call Algorithm LP_Interval, for solving the longest path problem on interval graphs; it consists of three phases and works as follows:

- Phase 1: it takes an interval graph G and constructs the auxiliary interval graph H;
- Phase 2: it computes a longest path P on H using Algorithm LP_on_H;
- Phase 3: it computes a longest path $\widehat{P}$ on G from the path P;

The proposed algorithm computes a longest path P of the graph H using dynamic programming techniques and, then, computes a longest path $\widehat{P}$ of G from the path P. We next describe in detail the three phases of our algorithm and prove properties of the constructed graph H which will be used for proving the correctness of the algorithm.

3.1 The interval graph H

In this section we present Phase 1 of the algorithm: given an interval graph G and a right-end ordering π of G, we construct the interval graph H and a right-end ordering σ of H.

- **Construction of** H **and** σ**:** Let G be an interval graph and let $\pi = (v_1, v_2, \ldots, v_{|V(G)|})$ be a right-end ordering of G. Initially, set $V(H) = V(G)$, $\sigma = \pi$, and $A = \emptyset$. Traverse the vertices of π from left to right and do the following: for every vertex v_i add two vertices $a_{i,1}$ and $a_{i,2}$ to the sets $V(H)$ and A, and make both these vertices to be adjacent to every vertex in $N_G[v_i] \cap \{v_i, v_{i+1}, \ldots, v_{|V(G)|}\}$. Update σ such that $a_{1,1} <_\sigma a_{1,2} <_\sigma v_1$, and $v_{i-1} <_\sigma a_{i,1} <_\sigma a_{i,2} <_\sigma v_i$ for every i, $2 \leq i \leq |V(G)|$.

We call the constructed graph H the *stable-connection graph* of the graph G. Hereafter, we will denote by n the number $|V(H)|$ of vertices of the graph H and by $\sigma = (u_1, u_2, \ldots, u_n)$ the constructed ordering of H. By construction, the vertex set of the graph H consists of the vertices of the set $C = V(G)$ and the vertices of the set A. We will refer to C as the set of the *connector vertices* c of the graph H and to A as the set of *stable vertices* a of the graph H; we denote these sets by $C(H)$ and $A(H)$, respectively. Note that $|A(H)| = 2|V(G)|$.

By the construction of the stable-connection graph H, all neighbors of a stable vertex $a \in A(H)$ are connector vertices $c \in C(H)$, such that $a <_\sigma c$. Moreover, observe that all neighbors of a stable vertex form a clique in G and, thus, also in H. For every connector vertex $u_i \in C(H)$, we denote by $u_{f(u_i)}$ and $u_{h(u_i)}$ the leftmost and rightmost neighbor of u_i in σ, respectively, which appear before u_i in σ, i.e., $u_{f(u_i)} <_\sigma u_{h(u_i)} <_\sigma u_i$. Note that $u_{f(u_i)}$ and $u_{h(u_i)}$ are distinct stable vertices, for every connector vertex u_i.

Lemma 5. *Let G be an interval graph. The stable-connection graph H of G is an interval graph, and the vertex ordering σ is a right-end ordering of H.*

Proof. Consider the intersection model F of G, from we each we obtain the right-end ordering $\pi = (v_1, v_2, \ldots, v_{|V(G)|})$ of G. Let I_i denote the interval which corresponds to the vertex v_i in F, and let $l(I_i)$ and $r(I_i)$ denote the left and the right endpoint of the interval I_i, respectively. Without loss of generality, we may assume that all values $l(I_i)$ and $r(I_i)$ are distinct. Let ε be the smallest distance between two interval endpoints in F.

For every interval I_i which corresponds to a vertex $v_i \in C$, we replace its right endpoint $r(I_i)$ by $r(I_i) + \frac{\varepsilon}{2}$, and we add two non-intersecting intervals $I_{i,1} = [r(I_i), r(I_i) + \frac{\varepsilon}{8}]$ and $I_{i,2} = [r(I_i) + \frac{\varepsilon}{4}, r(I_i) + \frac{3\varepsilon}{8}]$ (one for each vertex $a_{i,1}$ and $a_{i,2}$ of A, respectively). The two new intervals do not intersect with any interval I_k, such that $r(I_k) < r(I_i)$. Additionally, the two new intervals intersect with the interval I_i, and with every interval I_ℓ, such that $r(I_\ell) > r(I_i)$ and I_ℓ intersects with I_i. After processing all intervals I_i, $1 \leq i \leq |V(G)|$, of the intersection model F of G, we obtain an intersection model of H. Thus, H is an interval graph, and the ordering which results from numbering the intervals after sorting them on their right ends is identical to the vertex ordering σ of H and, thus, σ is a right-end ordering of H. ☐

Definition 2. *Let H be the stable-connection graph of an interval graph G, and let $\sigma = (u_1, u_2, \ldots, u_n)$ be the right-end ordering of H. For every pair of indices i, j, $1 \leq i \leq j \leq n$, we define the graph $H(i,j)$ to be the subgraph $H[S]$ of H, induced by the the set $S = \{u_i, u_{i+1}, \ldots, u_j\} \setminus \{u_k \in C(H) : u_{f(u_k)} <_\sigma u_i\}$.*

The following properties hold for every induced subgraph $H(i,j)$, $1 \leq i \leq j \leq n$, and they are used for proving the correctness of Algorithm LP_on_H.

Observation 1 *Let u_k be a connector vertex of $H(i,j)$, i.e., $u_k \in C(H(i,j))$. Then, for every vertex $u_\ell \in V(H(i,j))$, such that $u_k <_\sigma u_\ell$ and $u_k u_\ell \in E(H(i,j))$, u_ℓ is also a connector vertex of $H(i,j)$.*

Observation 2 *No two stable vertices of $H(i,j)$ are adjacent.*

Lemma 6. *Let $P = (v_1, v_2, \ldots, v_k)$ be a normal path of $H(i,j)$. Then:*

(a) For any two stable vertices v_r and v_ℓ in P, v_r appears before v_ℓ in P if and only if $v_r <_\sigma v_\ell$.
(b) For any two connector vertices v_r and v_ℓ in P, if v_ℓ appears before v_r in P and $v_r <_\sigma v_\ell$, then v_r does not see the previous vertex $v_{\ell-1}$ of v_ℓ in P.

Proof. The proof will be done by contradiction.

(a) Let v_r and v_ℓ be any two stable vertices of $H(i,j)$ that belong to the normal path $P = (v_1, v_2, \ldots, v_k)$, such that v_r appears before v_ℓ in P, and assume that $v_\ell <_\sigma v_r$. Then, clearly $v_\ell \neq v_1$, since v_r appears before v_ℓ in P. Since P is a normal path of $H(i,j)$, v_1 is the leftmost vertex of $V(P)$ in σ. Thus, $v_1 <_\sigma v_\ell <_\sigma v_r$, and since no two stable vertices of $H(i,j)$ are adjacent due to Observation 2, it follows that $v_r v_\ell \notin E(H(i,j))$. Thus, by Lemma 2 there exist two consecutive vertices u and u' in P that appear between v_1 and v_r in P, such that $uv_\ell \in E(H(i,j))$ and $v_\ell <_\sigma u'$. Thus, since P is a normal path, v_ℓ should be the next vertex of u in P instead of u', which is a contradiction. Therefore, $v_r <_\sigma v_\ell$.
(b) Let v_r and v_ℓ be any two connector vertices of $H(i,j)$ that belong to the normal path $P = (v_1, v_2, \ldots, v_k)$, such that v_ℓ appears before v_r in P and $v_r <_\sigma v_\ell$. Since P is a normal path of $H(i,j)$, v_1 is the leftmost vertex of $V(P)$ in σ. Since $v_r <_\sigma v_\ell$, it follows that $v_\ell \neq v_1$ and, thus, there exists a vertex $v_{\ell-1}$ which appears before v_ℓ in P. Assume that $v_r v_{\ell-1} \in E(H(i,j))$. Since $v_r <_\sigma v_\ell$, and since P is a normal path, v_r should be the next vertex of $v_{\ell-1}$ in P instead of v_ℓ, which is a contradiction. Therefore, $v_r v_{\ell-1} \notin E(H(i,j))$.

□

3.2 Finding a longest path on H

In this section we present Phase 2 of Algorithm LP_Interval. Let G be an interval graph and let H be the stable-connection graph of G constructed in Phase 1. We next present Algorithm LP_on_H, which computes a longest path of the graph H. Let us first give some definitions and notations necessary for the description of the algorithm.

Definition 3. *Let H be a stable-connection graph, and let P be a path of $H(i,j)$, $1 \leq i \leq j \leq n$. The path P is called* binormal *if P is a normal path of $H(i,j)$, both endpoints of P are stable vertices, and no two connector vertices are consecutive in P.*

Notation 1 *Let H be a stable-connection graph, and let $\sigma = (u_1, u_2, \ldots, u_n)$ be the right-end ordering of H. For every stable vertex $u_k \in A(H(i,j))$, we denote by $P(u_k; i, j)$ a longest binormal path of $H(i,j)$ with u_k as its right endpoint, and by $\ell(u_k; i, j)$ the length of $P(u_k; i, j)$.*

Algorithm LP_on_H

Input: a stable-connection graph H, a right-end ordering $\sigma = (u_1, u_2, \ldots, u_n)$ of H.

Output: a longest binormal path of H.

```
for j = 1 to n
    for i = j downto 1
       if i = j and u_i ∈ A(H) then
          ℓ(u_i; i, i) ← 1;  P(u_i; i, i) = (u_i);
       if i ≠ j then
          for every stable vertex u_k ∈ A(H), i ≤ k ≤ j − 1
             ℓ(u_k; i, j) ← ℓ(u_k; i, j − 1);  P(u_k; i, j) = P(u_k; i, j − 1);  {initialization}
          if u_j is a stable vertex of H(i, j), i.e., u_j ∈ A(H) then
             ℓ(u_j; i, j) ← 1;  P(u_j; i, j) = (u_j);
          if u_j is a connector vertex of H(i, j), i.e., u_j ∈ C(H) and i ≤ f(u_j) then
             execute process(H(i, j));
compute the max{ℓ(u_k; 1, n) : u_k ∈ A(H)} and the corresponding path P(u_k; 1, n);
```

where the procedure `process()` *is as follows:*

```
process(H(i, j))

for y = f(u_j) + 1 to j − 1
    for x = f(u_j) to y − 1      {u_x and u_y are adjacent to u_j}
       if u_x, u_y ∈ A(H) then
          w_1 ← ℓ(u_x; i, j − 1);  P'_1 = P(u_x; i, j − 1);
          w_2 ← ℓ(u_y; x + 1, j − 1);  P'_2 = P(u_y; x + 1, j − 1);
          if w_1 + w_2 + 1 > ℓ(u_y; i, j) then
             ℓ(u_y; i, j) ← w_1 + w_2 + 1;  P(u_y; i, j) = (P'_1, u_j, P'_2);
return the value ℓ(u_k; i, j) and the path P(u_k; i, j), for every vertex u_k ∈ A(H(f(u_j) + 1, j − 1));
```

Fig. 1. The algorithm for finding a longest binormal path of H.

Since any binormal path is a normal path, Lemma 6 also holds for binormal paths. Moreover, since $P(u_k; i, j)$ is a binormal path, it follows that its right endpoint u_k is also the rightmost stable vertex of P in σ, due to Lemma 6(a).

Algorithm LP_on_H, which is presented in Figure 1, computes for every induced subgraph $H(i, j)$ and for every stable vertex $u_k \in A(H(i, j))$, the length $\ell(u_k; i, j)$ and the corresponding path $P(u_k; i, j)$. Since $H(1, n) = H$, it follows that the maximum among the values $\ell(u_k; 1, n)$, where $u_k \in A(H)$, is the length of a longest binormal path $P(u_k; 1, n)$ of H. In Section 4.2 we prove that the length of a longest path of H equals to the length of a longest binormal path of H. Thus, the binormal path $P(u_k; 1, n)$ computed by Algorithm LP_on_H is also a longest path of H.

3.3 Finding a longest path on G

During Phase 3 of our Algorithm LP_Interval, we compute a path $\widehat{P}$ from the longest binormal path P of H, computed by Algorithm LP_on_H, by simply deleting all the stable vertices of P. In Section 4.2 we prove that the resulting path $\widehat{P}$ is a longest path of the interval graph G.

Algorithm LP_Interval

Input: an interval graph G and a right-end ordering π of G.

Output: a longest path $\widehat{P}$ of G.

1. Construct the stable-connection graph H of G and the right-end ordering σ of H;
   let $V(H) = C \cup A$, where $C = V(G)$ and A are the sets of the connector and stable vertices of H, respectively;
2. Compute a longest binormal path P of H, using Algorithm LP_on_H;
   let $P = (v_1, v_2, \ldots, v_{2k}, v_{2k+1})$, where $v_{2i} \in C$, $1 \leq i \leq k$, and $v_{2i+1} \in A$, $0 \leq i \leq k$;
3. Compute a longest path $\widehat{P} = (v_2, v_4, \ldots, v_{2k})$ of G, by deleting all stable vertices $\{v_1, v_3, \ldots, v_{2k+1}\}$ from the longest binormal path P of H;

Fig. 2. The algorithm for solving the longest path problem on an interval graph G.

In Figure 2, we present our Algorithm LP_Interval for solving the longest path problem on an interval graph G; note that Steps 1, 2, and 3 of the algorithm correspond to the presented Phases 1, 2, and 3, respectively.

4 Correctness and Time Complexity

In this section we prove the correctness of our algorithm and compute its time complexity. More specifically, in Section 4.1 we show that Algorithm LP_on_H computes a longest binormal path P of the graph H (in Lemma 13 we prove that this path is also a longest path of H), while in Section 4.2 we show that the length of a longest binormal path P of H is equal to $2k+1$, where k is the length of a longest path of G. Finally, we show that the path $\widehat{P}$ constructed at Step 3 of Algorithm LP_Interval is a longest path of G.

4.1 Correctness of Algorithm LP_on_H

We next prove that Algorithm LP_on_H correctly computes a longest binormal path of the graph H. The following lemmas appear useful in the proof of the algorithm's correctness.

Lemma 7. *Let H be a stable-connection graph, and let $\sigma = (u_1, u_2, \ldots, u_n)$ be the right-end ordering of H. Let P be a longest binormal path of $H(i,j)$ with u_y as its right endpoint, let u_k be the rightmost connector vertex of $H(i,j)$ in σ, and let $u_{f(u_k)+1} \leq_\sigma u_y \leq_\sigma u_{h(u_k)}$. Then, there exists a longest binormal path P' of $H(i,j)$ with u_y as its right endpoint, which contains the connector vertex u_k.*

Proof. Let P be a longest binormal path of $H(i,j)$ with u_y as its right endpoint, which does not contain the connector vertex u_k. Assume that $P = (u_y)$. Since u_k is a connector vertex of $H(i,j)$ and $u_{f(u_k)}$ is a stable vertex of $H(i,j)$, we have that $u_i \leq_\sigma u_{f(u_k)} <_\sigma u_y <_\sigma u_k$. Thus, there exists a binormal path $P_1 = (u_{f(u_k)}, u_k, u_y)$ such that $|P_1| > |P|$. However, this is a contradiction to the assumption that P is a longest binormal path of $H(i,j)$.

Therefore, assume now that $P = (u_p, \ldots, u_q, u_\ell, u_y)$. By assumption, P is a longest binormal path of $H(i,j)$ with u_y as its right endpoint that does not contain the connector vertex u_k. Since the connector vertex u_ℓ sees the stable vertex u_y and, also, since u_k is the rightmost connector vertex of $H(i,j)$ in σ, it follows by Observation 1 that $u_{f(u_k)} <_\sigma u_y <_\sigma u_\ell <_\sigma u_k$. Thus, u_k sees the connector vertex u_ℓ. Consider first the case where u_k does not see the stable vertex u_q, i.e., $u_q <_\sigma u_{f(u_k)} <_\sigma u_y <_\sigma u_\ell <_\sigma u_k$. Then, it is easy to see that the connector vertex u_ℓ sees $u_{f(u_k)}$, where $u_{f(u_k)}$ is always a stable vertex, and also, from Lemma 6(a) it follows that the vertex $u_{f(u_k)}$ does not belong to the path P. Therefore, there exists a binormal path $P_2 = (u_p, \ldots, u_q, u_\ell, u_{f(u_k)}, u_k, u_y)$ in $H(i,j)$, such that $|P_2| > |P|$. This is a contradiction to our assumption that P is a longest binormal path.

Consider now the case where u_k sees the stable vertex u_q. Then there exists a path $P' = (u_p, \ldots, u_q, u_k, u_y)$ of $H(i,j)$ with u_y as its right endpoint that contains the connector vertex u_k, such that $|P| = |P'|$; since P is a binormal path, it is easy to see that P' is also a binormal path. Thus, the path P' is a longest binormal path of $H(i,j)$ with u_y as its right endpoint, which contains the connector vertex u_k. □

Lemma 8. *Let H be a stable-connection graph, and let σ be the right-end ordering of H. Let $P = (P_1, v_\ell, P_2)$ be a binormal path of $H(i,j)$, and let v_ℓ be a connector vertex of $H(i,j)$. Then, P_1 and P_2 are binormal paths of $H(i,j)$.*

Proof. Let $P = (v_1, v_2, \ldots, v_{\ell-1}, v_\ell, v_{\ell+1}, \ldots, v_k)$ be a binormal path of $H(i,j)$. Then, from Definition 1, v_1 is the leftmost vertex of $V(P)$ in σ, and for every index r, $2 \leq r \leq k$, the vertex v_r is the leftmost vertex of $N(v_{r-1}) \cap \{v_r, v_{r+1}, \ldots, v_k\}$ in σ. It is easy to see that $P_1 = (v_1, v_2, \ldots, v_{\ell-1})$ is a normal path of $H(i,j)$. Indeed, since $V(P_1) \subset V(P)$, then v_1 is also the leftmost vertex of $V(P_1)$ in σ, and additionally, v_r is the leftmost vertex of $N(v_{r-1}) \cap \{v_r, v_{r+1}, \ldots, v_{\ell-1}\}$ in σ, for every index r, $2 \leq r \leq \ell - 1$. Furthermore, since P is binormal and v_ℓ is a connector vertex, it follows that $v_{\ell-1}$ is a stable vertex and, thus, P_1 is a binormal path of $H(i,j)$ as well.

Consider now the path $P_2 = (v_{\ell+1}, v_{\ell+2}, \ldots, v_k)$ of $H(i,j)$. Since P is a binormal path and v_ℓ is a connector vertex, it follows that $v_{\ell+1}$ is a stable vertex and, thus, $v_{\ell+1} <_\sigma v_\ell$ due to Observation 1. We first prove that $v_{\ell+1}$ is the leftmost vertex of $V(P_2)$ in σ. Since P is a binormal path, we obtain from Lemma 6(a) that $v_{\ell+1}$ is the leftmost stable vertex of $V(P_2)$ in σ. Moreover, consider a connector vertex v_t of P_2. Then, its previous vertex v_{t-1} in P_2 is a stable vertex and, thus, $v_{t-1} <_\sigma v_t$ due to Observation 1. Since $v_{\ell+1}$ is the leftmost stable vertex of $V(P_2)$ in σ, we have that $v_{\ell+1} \leq_\sigma v_{t-1}$ and, thus, $v_{\ell+1} <_\sigma v_t$. Therefore, $v_{\ell+1}$ is the leftmost vertex of $V(P_2)$ in σ. Additionally, since P is a binormal path, it is straightforward that for every index r, $\ell+2 \leq r \leq k$, the vertex v_r is the leftmost vertex of $N(v_{r-1}) \cap \{v_r, v_{r+1}, \ldots, v_k\}$ in σ. Thus, P_2 is a normal path. Finally, since P is binormal and $v_{\ell+1}$ is a stable vertex, P_2 is a binormal path as well. □

Lemma 9. *Let H be a stable-connection graph, and let $\sigma = (u_1, u_2, \ldots, u_n)$ be the right-end ordering of H. Let P_1 be a binormal path of $H(i, j-1)$ with u_x as its right endpoint, and let P_2 be a binormal path of $H(x+1, j-1)$ with u_y as its right endpoint, such that $V(P_1) \cap V(P_2) = \emptyset$. Suppose that u_j is a connector vertex of H and that $u_i \leq_\sigma u_{f(u_j)} \leq_\sigma u_x$. Then $P = (P_1, u_j, P_2)$ is a binormal path of $H(i,j)$ with u_y as its right endpoint.*

Proof. Let $P_1 = (v_1, v_2, \ldots, v_{p-1})$, $P_2 = (v_{p+1}, v_{p+2}, \ldots, v_\ell)$, and $P = (P_1, u_j, P_2) = (v_1, v_2, \ldots, v_{p-1}, v_p, v_{p+1}, v_{p+2}, \ldots, v_\ell)$, where $v_p = u_j$ is a connector vertex of H and

$u_i \leq_\sigma u_{f(u_j)} \leq_\sigma u_x$. On the one hand, $u_{f(u_j)} \leq_\sigma u_x = v_{p-1}$ and, thus, $v_p = u_j$ sees v_{p-1}. On the other hand, since $v_{p+1} \in V(H(x+1, j-1))$, we have $u_{f(u_j)} \leq_\sigma u_x <_\sigma u_{x+1} \leq_\sigma v_{p+1} <_\sigma u_j$ and, thus, $v_p = u_j$ sees v_{p+1}. Therefore, since $V(P_1) \cap V(P_2) = \emptyset$, it follows that P is a path of H. Additionally, since $H(i, j-1)$ and $H(x+1, j-1)$ are induced subgraphs of $H(i,j)$, it follows that P is a path of $H(i,j)$.

We first show that $P = (v_1, v_2, \ldots, v_p, \ldots, v_\ell)$ is a normal path. Since v_1 is the leftmost vertex of $V(P_1)$ in σ, it follows that $v_1 \leq_\sigma u_x$. Furthermore, since for every vertex $v_k \in V(P_2)$ it holds $u_x <_\sigma u_{x+1} \leq_\sigma v_k$, it follows that v_1 is the leftmost vertex of $V(P)$ in σ. We next show that for every k, $2 \leq k \leq \ell$, the vertex v_k is the leftmost vertex of $N(v_{k-1}) \cap \{v_k, v_{k+1}, \ldots, v_\ell\}$ in σ.

Consider first the case where $2 \leq k \leq p-1$, i.e., $v_k \in V(P_1)$. Since P_1 is a normal path, v_k is the leftmost vertex of $N(v_{k-1}) \cap \{v_k, v_{k+1}, \ldots, v_{p-1}\}$ in σ. Assume that v_{k-1} is a stable vertex. Then, Lemma 6(a) implies that $v_{k-1} <_\sigma v_{p-1} = u_x$ and, due to Observation 2, it follows that $N(v_{k-1}) \cap \{v_k, v_{k+1}, \ldots, v_\ell\}$ is a set of connector vertices. Since every connector vertex $v_r \in V(P_2)$ is a vertex of $H(x+1, j-1)$, it follows that $v_{k-1} <_\sigma u_{x+1} \leq_\sigma u_{f(v_r)}$ and, thus, $v_r \notin N(v_{k-1})$. Additionally, since $v_p = u_j$ is the rightmost vertex of $H(i,j)$ in σ, it follows that $v_k <_\sigma v_p$. Therefore, since v_k is the leftmost vertex of $N(v_{k-1}) \cap \{v_k, v_{k+1}, \ldots, v_{p-1}\}$ in σ, it follows that v_k is the leftmost vertex of $N(v_{k-1}) \cap \{v_k, v_{k+1}, \ldots, v_\ell\}$ in σ. Assume now that v_{k-1} is a connector vertex. Since P_1 is a binormal path, v_k is a stable vertex, such that $v_k \leq_\sigma u_x$ and v_k is the leftmost vertex of $N(v_{k-1}) \cap \{v_k, v_{k+1}, \ldots, v_{p-1}\}$ in σ. Since for every r, $p+1 \leq r \leq \ell$, the vertex $v_r \in V(H(x+1, j-1))$, it follows that $v_k \leq_\sigma u_x <_\sigma v_r$. Additionally, $v_k <_\sigma u_{x+1} <_\sigma v_p$. Therefore, v_k is the leftmost vertex of $N(v_{k-1}) \cap \{v_k, v_{k+1}, \ldots, v_\ell\}$ in σ.

Consider now the case where $k = p$. Since P_1 is a normal path and $v_{p-1} = u_x$ is a stable vertex, $N(v_{p-1}) \cap \{v_p, v_{p+1}, \ldots, v_\ell\}$ is a set of connector vertices, due to Observation 2. Additionally, since every connector vertex $v_r \in V(P_2)$ is a vertex of $H(x+1, j-1)$, it follows that $v_{p-1} <_\sigma u_{x+1} \leq_\sigma u_{f(v_r)}$ and, thus, $v_r \notin N(v_{p-1})$. Therefore, $N(v_{p-1}) \cap \{v_p, v_{p+1}, \ldots, v_\ell\} = \{v_p\}$ and, thus, v_p is the leftmost vertex of $N(v_{p-1}) \cap \{v_p, v_{p+1}, \ldots, v_\ell\}$ in σ. Now, in the case where $k = p+1$, we have that v_{p+1} is the leftmost vertex of $V(P_2) = \{v_{p+1}, v_{p+2}, \ldots, v_\ell\}$ in σ, since P_2 is a normal path. Therefore, it easily follows that v_{p+1} is the leftmost vertex of $N(v_p) \cap \{v_{p+1}, v_{p+2}, \ldots, v_\ell\}$ in σ. Finally, in the case where $p+2 \leq k \leq \ell$, since P_2 is a normal path it directly follows that v_k is the leftmost vertex of $N(v_{k-1}) \cap \{v_k, v_{k+1}, \ldots, v_\ell\}$ in σ.

Concluding, we have shown that P is a normal path of $H(i,j)$. Additionally, since P_1 and P_2 are binormal paths of $H(i,j)$, the path P has stable vertices as endpoints and no two connector vertices are consecutive in P. Therefore, P is a binormal path of $H(i,j)$ with u_y as its right endpoint. □

Next, we prove the correctness of Algorithm LP_on_H.

Lemma 10. *Let H be a stable-connection graph, and let σ be the right-end ordering of H. For every induced subgraph $H(i,j)$ of H, $1 \leq i \leq j \leq n$, and for every stable vertex $u_y \in A(H(i,j))$, Algorithm LP_on_H computes the length $\ell(u_y; i, j)$ of a longest binormal path of $H(i,j)$ which has u_y as its right endpoint and, also, the corresponding path $P(u_y; i, j)$.*

Proof. Let P be a longest binormal path of the stable-connection graph $H(i,j)$, which has a vertex $u_y \in A(H(i,j))$ as its right endpoint. Consider first the case where $C(H(i,j)) = \emptyset$; the graph $H(i,j)$ is consisted of a set of stable vertices $A(H(i,j))$, which is an independent set, due to Observation 2. Therefore, in this case Algorithm LP_on_H sets $\ell(u_y;i,j) = 1$ for every vertex $u_y \in A(H(i,j))$, which is indeed the length of the longest binormal path $P(u_y;i,j) = (u_y)$ of $H(i,j)$ which has u_y as its right endpoint. Therefore, the lemma holds for every induced subgraph $H(i,j)$, for which $C(H(i,j)) = \emptyset$.

We examine next the case where $C(H(i,j)) \neq \emptyset$. Let $C(H) = \{c_1, c_2, \ldots, c_k, \ldots, c_t\}$ be the set of connector vertices of H, where $c_1 <_\sigma c_2 <_\sigma \ldots <_\sigma c_k <_\sigma \ldots <_\sigma c_t$. Let $\sigma = (u_1, u_2, \ldots, u_n)$ be the vertex ordering of H constructed in Phase 1. Recall that, by the construction of H, $n = 3t$, and $A(H) = V(H) \setminus C(H)$ is the set of stable vertices of H.

Let $H(i,j)$ be an induced subgraph of H, and let c_k be the rightmost connector vertex of $H(i,j)$ in σ. The proof of the lemma is done by induction on the index k of the rightmost connector vertex c_k of $H(i,j)$. More specifically, given a connector vertex c_k of H, we prove that the lemma holds for every induced subgraph $H(i,j)$ of H, which has c_k as its rightmost connector vertex in σ. To this end, in both the induction basis and the induction step, we distinguish three cases on the position of the stable vertex u_y in the ordering σ: $u_i \leq_\sigma u_y \leq_\sigma u_{f(c_k)}$, $u_{h(c_k)} <_\sigma u_y \leq_\sigma u_j$, and $u_{f(c_k)+1} \leq_\sigma u_y \leq_\sigma u_{h(c_k)}$. In each of these three cases, we examine first the length of a longest binormal path of $H(i,j)$ with u_y as its right endpoint and, then, we compare this value to the length of the path computed by Algorithm LP_on_H. Moreover, we prove that the path computed by Algorithm LP_on_H is a binormal path with u_y as its right endpoint.

We first show that the lemma holds for $k = 1$. In the case where $u_i \leq_\sigma u_y \leq_\sigma u_{f(c_1)}$ or $u_{h(c_1)} <_\sigma u_y \leq_\sigma u_j$, it is easy to see that the length $\ell(u_y;i,j)$ of a longest binormal path P of $H(i,j)$ with u_y as its right endpoint is equal to 1. Indeed, in these cases, if $u_y \neq u_{f(c_1)}$, then u_y does not see the unique connector vertex c_1 of $H(i,j)$ and, thus, the longest binormal path with u_y as its right endpoint is consisted of the vertex u_y. Now, in the case where $u_y = u_{f(c_1)}$, the connector vertex c_1 sees u_y, however, c_1 does not belong to any binormal path with u_y as its right endpoint, since u_y is the leftmost neighbor of c_1 in σ. Therefore, in the case where $u_i \leq_\sigma u_y \leq_\sigma u_{f(c_1)}$ or $u_{h(c_1)} <_\sigma u_y \leq_\sigma u_j$, Algorithm LP_on_$H$ computes the length of the longest binormal path $P(u_y;i,j) = (u_y)$ of $H(i,j)$ with u_y as its right endpoint. In the case where $u_{f(c_1)+1} \leq_\sigma u_y \leq_\sigma u_{h(c_1)}$, Algorithm LP_on_$H$ computes (in the subroutine `process()`) for every stable vertex u_x of $H(i,j)$, such that $u_{f(c_1)} \leq_\sigma u_x \leq_\sigma u_{y-1}$, the value $\ell(u_x;i,j-1) + \ell(u_y;x+1,j-1) + 1 = 1 + 1 + 1 = 3$ and sets $\ell(u_y;i,j) = 3$. It is easy to see that the path $P(u_y;i,j) = (u_x, c_1, u_y)$, computed by Algorithm LP_on_H in this case, is indeed a longest binormal path of $H(i,j)$ with u_y as its right endpoint.

Let now c_k be a connector vertex of H, such that $k \leq t$. Assume that the lemma holds for every induced subgraph $H(i,j)$ of H, which has c_ℓ as its rightmost connector vertex in σ, where $1 \leq \ell \leq k-1$. That is, we assume that for every such graph $H(i,j)$, the value $\ell(u_y;i,j)$ computed by Algorithm LP_on_H is the length of a longest binormal path $P(u_y;i,j)$ of $H(i,j)$ with u_y as its right endpoint. We will show that the lemma holds for every induced subgraph $H(i,j)$ of H, which has c_k as its rightmost connector vertex in σ.

Case 1: $u_i \leq_\sigma u_y \leq_\sigma u_{f(c_k)}$. In this case, it holds $\ell(u_y;i,j) = \ell(u_y;i,h(c_k))$ (note that $u_{h(c_k)}$ is the previous vertex of c_k in σ). Indeed, on the one hand, using similar arguments as in the induction basis, it easily follows that the connector vertex c_k does not belong to any binormal path of $H(i,j)$ with u_y as its right endpoint. On the other hand, since c_k is the rightmost connector vertex of $H(i,j)$, it follows that every vertex u_ℓ of $H(i,j)$, where $c_k <_\sigma u_\ell \leq_\sigma u_j$, is a stable vertex and, thus, u_ℓ does not see u_y, due to Observation 2. Therefore, we obtain that $\ell(u_y;i,j) = \ell(u_y;i,h(c_k))$.

Next, we show that this is the result computed by Algorithm LP_on_H in this case. First, we show that Algorithm LP_on_H has already computed the value $\ell(u_y;i,h(c_k))$, which is the length of a longest binormal path $P(u_y;i,h(c_k))$ of $H(i,h(c_k))$ with u_y as its right endpoint. Indeed, in the case where $H(i,h(c_k))$ is a graph for which $C(H(i,h(c_k))) = \emptyset$, it is easy to see that Algorithm LP_on_H has already computed the value $\ell(u_y;i,h(c_k))$ in the iteration where j was equal to $h(c_k)$. Consider now the case where $H(i,h(c_k))$ is a graph for which $C(H(i,h(c_k))) \neq \emptyset$. If c_ℓ is the rightmost connector vertex of $H(i,h(c_k))$ in σ, it follows that $c_\ell <_\sigma c_k$, since $u_{h(c_k)} <_\sigma c_k$. Therefore, by the induction hypothesis, Algorithm LP_on_H has already computed the length $\ell(u_y;i,h(c_k))$ of a longest binormal path of the graph $H(i,h(c_k))$ with u_y as its right endpoint and, also, the corresponding path $P(u_y;i,h(c_k))$.

We now show that in Case 1 Algorithm LP_on_H computes $\ell(u_y;i,j) = \ell(u_y;i,h(c_k))$. Indeed, in the case where u_j is a connector vertex of $H(i,j)$, i.e., $u_j = c_k$, Algorithm LP_on_H computes $\ell(u_y;i,j) = \ell(u_y;i,j-1)$, which equals to $\ell(u_y;i,h(c_k))$, since in this case $j-1 = h(c_k)$. In the case where u_j is a stable vertex, then again Algorithm LP_on_H computes $\ell(u_y;i,j) = \ell(u_y;i,j-1)$, which is again equal to $\ell(u_y;i,h(c_k))$, since the vertex $u_{h(c_k)+1} = c_k$ does not belong to any binormal path of $H(i,j)$ with u_y as its right endpoint, and since every vertex u_ℓ, such that $u_{h(c_k)+1} <_\sigma u_\ell \leq_\sigma u_j$, is a stable vertex and does not see u_y. Therefore, in the case where $u_i \leq_\sigma u_y \leq_\sigma u_{f(c_k)}$, Algorithm LP_on_$H$ computes $\ell(u_y;i,h(c_k))$ as the length of a longest path of $H(i,j)$ with u_y as its right endpoint and, also, computes $P(u_y;i,j) = P(u_y;i,h(c_k))$. Then, by the induction hypothesis, this path is binormal. Thus, in Case 1 the lemma holds.

Case 2: $u_{h(c_k)} <_\sigma u_y \leq_\sigma u_j$. Since c_k is the rightmost connector vertex of $H(i,j)$, and since u_y is a stable vertex, it follows that u_y does not see any vertex of $H(i,j)$. Thus, the longest binormal path of $H(i,j)$ with u_y as its right endpoint is consisted of the vertex u_y, i.e., $\ell(u_y;i,j) = 1$. One can easily see that in this case Algorithm LP_on_H computes the length $\ell(u_y;i,j) = 1$, and the path $P(u_y;i,j) = (u_y)$, which is clearly a binormal path. Thus, in Case 2 the lemma holds.

Case 3: $u_{f(c_k)+1} \leq_\sigma u_y \leq_\sigma u_{h(c_k)}$. In this case, the connector vertex c_k sees u_y. Let $P = (u_{x'}, \ldots, u_x, c_k, u_{y'}, \ldots, u_y)$ be a longest binormal path of $H(i,j)$ with u_y as its right endpoint, which contains the connector vertex c_k; due to Lemma 7, such a path always exists. Let u_x be the previous vertex of c_k in the path P; thus, $u_{f(c_k)} \leq_\sigma u_x <_\sigma u_y$. Since P is a binormal path, the vertices $u_{x'}$, u_x, $u_{y'}$, and u_y are all stable vertices. Also, since c_k sees u_y, which is the rightmost stable vertex of P in σ, all stable vertices of P belong to the graph $H(i,h(c_k))$. Additionally, since c_k is the rightmost connector vertex of $H(i,j)$ in σ, all connector vertices of P belong to the graph $H(i,h(c_k)+1)$. Therefore, all vertices of P belong to the graph $H(i,h(c_k)+1)$. Thus, the path P is a longest binormal path of $H(i,h(c_k)+1)$ with u_y as its right endpoint, which contains the connector vertex c_k. Therefore, for every graph $H(i,j)$, for which c_k is its rightmost connector vertex in σ and $h(c_k)+1 \leq j$, we have that $\ell(u_y;i,j) = \ell(u_y;i,h(c_k)+1)$. Thus, we will examine only the case where $h(c_k)+1 = j$, that is, c_k is the rightmost vertex u_j of $H(i,j)$ in σ.

Next, we examine the length $\ell(u_y; i, j)$ of a longest binormal path of $H(i, j)$ with u_y as its right endpoint, in the case where $h(c_k) + 1 = j$. Consider removing the connector vertex c_k from the path P. Then we obtain the paths $P_1 = (u_{x'}, \ldots, u_x)$ and $P_2 = (u_{y'}, \ldots, u_y)$. Since P is a binormal path of $H(i, j)$, from Lemma 8 we obtain that P_1 and P_2 are binormal paths of $H(i, j)$. Since, as we have shown, all vertices of P belong to $H(i, h(c_k) + 1)$, and since $c_k = u_j$ is the rightmost vertex of $H(i, j)$ in σ, it follows that all vertices of P_1 and P_2 belong to the graph $H(i, h(c_k)) = H(i, j - 1)$. Since P is a binormal path, from Lemma 6(a) it follows that for every stable vertex $u_{\ell_1} \in V(P_1)$, we have $u_i \leq_\sigma u_{x'} \leq_\sigma u_{\ell_1} \leq_\sigma u_x$. Additionally, for every stable vertex $u_{\ell_2} \in V(P_2)$, we have $u_x <_\sigma u_{\ell_2} \leq_\sigma u_y \leq_\sigma u_{j-1}$, where $u_{j-1} = u_{h(c_k)}$ is the rightmost vertex of $H(i, j - 1)$ in σ, since $u_j = c_k$. Therefore, for every stable vertex $u_{\ell_1} \in V(P_1)$ it holds $u_{\ell_1} \in A(H(i, x))$, and for every stable vertex $u_{\ell_2} \in V(P_2)$ it holds $u_{\ell_2} \in A(H(x + 1, j - 1))$.

Similarly, since P_1 is a binormal path, u_x is the rightmost stable vertex of $V(P_1)$ in σ, due to Lemma 6(a). Moreover, since P_1 is binormal, every connector vertex $c_{\ell_1} \in V(P_1)$ sees at least two stable vertices of P_1 and, thus, $u_i \leq_\sigma u_{f(c_{\ell_1})} <_\sigma u_x$. Therefore, for every connector vertex $c_{\ell_1} \in V(P_1)$, we have that $c_{\ell_1} \in C(H(i, j-1)) \setminus \{c_\ell \in C(H(i, j-1)) : u_x \leq_\sigma u_{f(c_\ell)}\} \subseteq C(H(i, j - 1)) \setminus C(H(x + 1, j - 1))$. Additionally, from Lemma 6(b) we have that every connector vertex $c_{\ell_2} \in V(P_2)$ does not see the vertex u_x, i.e., $u_x <_\sigma u_{f(c_{\ell_2})} <_\sigma c_{\ell_2} \leq_\sigma u_{j-1}$; thus, $c_{\ell_2} \in C(H(x + 1, j - 1))$. Summarizing, let H_1 and H_2 be the induced subgraphs of $H(i, j - 1)$, with vertex sets $V(H_1) = A(H(i, x)) \cup C(H(i, j - 1)) \setminus C(H(x + 1, j - 1))$ and $V(H_2) = A(H(x + 1, j - 1)) \cup C(H(x + 1, j - 1))$, respectively. Note that, the graphs H_1 and H_2 are defined with respect to a stable vertex u_x, where $u_{f(c_k)} \leq_\sigma u_x <_\sigma u_{j-1}$, and that $H_2 = H(x + 1, j - 1)$. Now, it is easy to see that $V(H_1) \cap V(H_2) = \emptyset$. Moreover, P_1 and P_2 belong to the graphs H_1 and H_2, respectively and, therefore, $V(P_1) \cap V(P_2) = \emptyset$.

Since $P = (P_1, c_k, P_2)$ is a longest binormal path of $H(i, j)$ with u_y as its right endpoint, and since the paths P_1 and P_2 belong to two disjoint induced subgraphs of $H(i, j)$, it follows that P_1 is a longest binormal path of H_1 with u_x as its right endpoint, and that P_2 is a longest binormal path of H_2 with u_y as its right endpoint. Thus, since $H_2 = H(x + 1, j - 1)$, we obtain that $|P_2| = \ell(u_y; x + 1, j - 1)$. We will now show that $|P_1| = \ell(u_x; i, j - 1)$. To this end, consider a longest binormal path P_0 of $H(i, j - 1)$ with u_x as its right endpoint. Due to Lemma 6(a), u_x is the rightmost stable vertex of P_0 in σ and, thus, all stable vertices of P_0 belong to $A(H_1) = A(H(i, x))$. Furthermore, since P_0 is binormal, every connector vertex c_ℓ of P_0 sees at least two stable vertices of P_0 and, thus, $u_{f(c_\ell)} <_\sigma u_x$, i.e., $c_\ell \in C(H_1) = C(H(i, j - 1)) \setminus C(H(x + 1, j - 1))$. It follows that $V(P_0) \subseteq V(H_1)$ and, thus, $|P_0| \leq |P_1|$. On the other hand, $|P_1| \leq |P_0|$, since H_1 is an induced subgraph of $H(i, j - 1)$. Thus, $|P_1| = |P_0| = \ell(u_x; i, j - 1)$. Therefore, for the length $|P| = \ell(u_y; i, j)$ of a longest binormal path P of $H(i, j)$ with u_y as its right endpoint, it follows that $\ell(u_y; i, j) = \ell(u_x; i, j - 1) + \ell(u_y; x + 1, j - 1) + 1$.

Hereafter, we examine the results computed by Algorithm LP_on_H in Case 3. Let P' be the path of the graph $H(i, j)$ with u_y as its right endpoint computed by Algorithm LP_on_H, in the case where $u_{f(c_k)+1} \leq_\sigma u_y \leq_\sigma u_{h(c_k)}$. Consider first the case where u_j is a connector vertex of $H(i, j)$, i.e., $u_j = c_k$. It is easy to see that the path P' constructed by Algorithm LP_on_H (in the subroutine `process()`) contains the connector vertex c_k. Algorithm LP_on_H computes the length of the path $P' = (P'_1, c_k, P'_2)$, for two paths P'_1 and P'_2 as follows. The path $P'_1 = P(u_x; i, j - 1)$ is a path of $H(i, j - 1)$ with u_x as its right endpoint, where u_x is a neighbor of c_k, such that $u_{f(c_k)} \leq_\sigma u_x <_\sigma u_y$. The path $P'_2 = P(u_y; x+1, j-1)$ is a path of $H(x+1, j-1)$ with u_y as its right endpoint, where $u_{f(c_k)+1} \leq_\sigma u_y \leq_\sigma u_{h(c_k)}$. Actually, in this case, Algorithm LP_on_H computes (in the subroutine `process()`) the value $w_1 + w_2 + 1 = |P'_1| + |P'_2| + 1$, for every stable vertex u_x, where $u_{f(c_k)} \leq_\sigma u_x <_\sigma u_y$, and sets $|P'|$ to be equal to the maximum among these values. Additionally, Algorithm LP_on_H computes the corresponding path $P' = (P'_1, c_k, P'_2)$.

By the induction hypothesis, Algorithm LP_on_H has already computed the values $|P'_1| = w_1$ and $|P'_2| = w_2$. Indeed, in the case where $H(i, j-1)$ is a graph for which $C(H(i, j-1)) = \emptyset$, it is easy to see that Algorithm LP_on_H has already computed the values $|P'_1| = w_1 = 1$ and $|P'_2| = w_2 = 1$. Consider now the case where $H(i, j-1)$ is a graph for which $C(H(i, j-1)) \neq \emptyset$. If c_ℓ is the rightmost connector vertex of the graph $H(i, j-1)$ in σ, it follows that $c_\ell <_\sigma c_k$, since $c_k = u_j$. Therefore, by the induction hypothesis, Algorithm LP_on_H has already computed the values $|P'_1| = \ell(u_x; i, j-1)$ and $|P'_2| = \ell(u_y; x+1, j-1)$. Thus, Algorithm LP_on_$H$ computes (in the subroutine `process()`), for every stable vertex u_x, where $u_{f(c_k)} \leq_\sigma u_x <_\sigma u_y$, the value $\ell(u_x; i, j-1) + \ell(u_y; x+1, j-1) + 1$, and sets $|P'|$ to be equal to the maximum among these values.

Since by the induction hypothesis, P'_1 and P'_2 are binormal paths of $H(i, j-1)$ with u_x and u_y as their right endpoints, respectively, it follows similarly to the above that P'_1 and P'_2 belong to the graphs H_1 and H_2, respectively. Recall that, the graphs H_1 and H_2 are defined with respect to a stable vertex u_x, where $u_{f(c_k)} \leq_\sigma u_x <_\sigma u_{j-1}$. Since, as we have shown, $V(H_1) \cap V(H_2) = \emptyset$, it follows that $V(P'_1) \cap V(P'_2) = \emptyset$. Therefore, from Lemma 9 we obtain that the computed path $P' = (P'_1, u_j, P'_2)$ is a binormal path as well and, thus, P' is a longest binormal path of $H(i, j)$ with u_y as its right endpoint.

Consider now the case where u_j is a stable vertex of $H(i, j)$. Let c_k be the rightmost connector vertex of $H(i, j)$ in σ; then $h(c_k) + 1 < j$. Assume first that $h(c_k) + 1 = j - 1$. Since u_j is a stable vertex and also the rightmost vertex of $H(i, j)$, u_j does not see any vertex of $H(i, h(c_k) + 1)$. In this case, Algorithm LP_on_H correctly computes the path $P' = P(u_y; i, j-1) = P(u_y; i, h(c_k) + 1)$, with length $|P'| = \ell(u_y; i, h(c_k) + 1)$. Similarly, in the case where $h(c_k) + 1 < j - 1$, Algorithm LP_on_H computes the path $P' = P(u_y; i, j-1) = P(u_y; i, h(c_k) + 1)$, with length $|P'| = \ell(u_y; i, j-1) = \ell(u_y; i, h(c_k) + 1)$. Algorithm LP_on_$H$ has already computed the value $\ell(u_y; i, h(c_k) + 1)$ at a previous iteration where j was equal to $h(c_k)+1$ (i.e., $u_j = c_k$) and, also, the computed path $P' = P(u_y; i, h(c_k)+1)$ is binormal.

Concluding, in both cases where u_j is a connector or a stable vertex of $H(i, j)$, the path P' of $H(i, j)$ with u_y as its right endpoint computed by Algorithm LP_on_H is a longest binormal path $P(u_y; i, j)$ of $H(i, j)$ with u_y as its right endpoint, and $|P'| = \ell(u_y; i, j)$. Thus, the lemma holds in Case 3 as well. □

Due to Lemma 10, and since the output of Algorithm LP_on_H is the maximum among the lengths $\ell(u_y; 1, n)$, $u_y \in A(H(1, n))$, along with the corresponding path, it follows that Algorithm LP_on_H computes a longest binormal path of $H(1, n)$ with right endpoint a vertex $u_y \in A(H(1, n))$. Thus, since $H(1, n) = H$, we obtain the following result.

Lemma 11. *Let G be an interval graph. Algorithm LP_on_H computes a longest binormal path of the stable-connection graph H of the graph G.*

4.2 Correctness of Algorithm LP_Interval

We next show that Algorithm LP_Interval correctly computes a longest path of an interval graph G. The correctness proof is based on the following property: for any longest path P of G there exists a longest binormal path P' of H, such that $|P'| = 2|P| + 1$ and vice versa (this property is proved in Lemma 12). Therefore, we obtain that the length of a longest binormal path P of H computed by Algorithm LP_on_H, is equal to $2k + 1$, where k is the length of a longest path $\widehat{P}$ of G. Next, we show that the length of a longest binormal path of H equals to the length of a longest path of H. Finally, we show that the path $\widehat{P}$ computed at Step 3 of Algorithm LP_Interval is indeed a longest path of the interval graph G.

Lemma 12. *Let H be the stable-connection graph of an interval graph G. Then, for any longest path P of G there exists a longest binormal path P' of H, such that $|P'| = 2|P| + 1$ and vice versa.*

Proof. Let σ be the right-end ordering of H, constructed in Phase 1.

($\Longrightarrow$) Let $P = (v_1, v_2, \ldots, v_k)$ be a longest path of G, i.e., $|P| = k$. We will show that there exists a binormal path P' of H such that $|P'| = 2k + 1$. Since G is an induced subgraph of H, the path P of G is a path of H as well. We construct a path $\widehat{P}$ of H from P, by adding to P the appropriate stable vertices, using the following procedure. Initially, set $\widehat{P} = P$ and for every subpath (v_i, v_{i+1}) of the path $\widehat{P}$, $1 \leq i \leq k - 1$, do the following: consider first the case where $v_i <_\sigma v_{i+1}$; then, by the construction of H, v_{i+1} is adjacent to both stable vertices $a_{i,1}$ and $a_{i,2}$ associated with the connector vertex v_i. If $a_{i,1}$ has not already been added to $\widehat{P}$, then replace the subpath (v_i, v_{i+1}) by the path $(v_i, a_{i,1}, v_{i+1})$; otherwise, replace the subpath (v_i, v_{i+1}) by the path $(v_i, a_{i,2}, v_{i+1})$. Similarly, in the case where $v_{i+1} <_\sigma v_i$, replace the subpath (v_i, v_{i+1}) by the path $(v_i, a_{i+1,1}, v_{i+1})$ or $(v_i, a_{i+1,2}, v_{i+1})$, respectively. Finally, consider the endpoint v_1 (resp. v_k) of $\widehat{P}$. If $a_{1,1}$ (resp. $a_{k,1}$) has not already been added to $\widehat{P}$, then add $a_{1,1}$ (resp. $a_{k,1}$) as the first (resp. last) vertex of $\widehat{P}$; otherwise, add $a_{1,2}$ (resp. $a_{k,2}$) as the first (resp. last) vertex of $\widehat{P}$.

By the construction of $\widehat{P}$ it is easy to see that for every connector vertex v of P we add two stable vertices as neighbors of v in $\widehat{P}$, and since in H there are exactly two stable vertices associated with every connector vertex v, it follows that every stable vertex of H appears at most once in $\widehat{P}$. Furthermore, since we add in total $k + 1$ stable vertices to P, where $|P| = k$, it follows that $|\widehat{P}| = 2k + 1$. Denote now by P' a normal path of H such that $V(P') = V(\widehat{P})$. Such a path exists, due to Lemma 4. Due to the above construction, the path $\widehat{P}$ is consisted of $k + 1$ stable vertices and k connector vertices. Thus, since no two stable vertices are adjacent in H due to Observation 2, and since P' is a normal path of H, it follows that P' is a binormal path of H. Thus, for any longest path P of G there exists a binormal path P' of H, such that $|P'| = 2|P| + 1$.

($\Longleftarrow$) Consider now a longest binormal path $P' = (v_1, v_2, \ldots, v_\ell)$ of H. Since P' is binormal, it follows that $\ell = 2k + 1$, and that P' has k connector vertices and $k + 1$ stable vertices, for some $k \geq 1$. We construct a path P by deleting all stable vertices from the path P' of H. By the construction of H, all neighbors of a stable vertex a are connector vertices and form a clique in G; thus, for every subpath (v, a, v') of P', v is adjacent to v' in G. It follows that P is a path of G. Since we removed all the $k + 1$ stable vertices of P', it follows that $|P| = k$, i.e., $|P'| = 2|P| + 1$.

Summarizing, we have constructed a binormal path P' of H from a longest path P of G such that $|P'| = 2|P| + 1$, and a path P of G from a longest binormal path P' of H such that $|P'| = 2|P| + 1$. This completes the proof. □

In the next lemma, we show that the length of a longest path of H is equal to the length of a longest binormal path of H.

Lemma 13. *For any longest path P and any longest binormal path P' of H, it holds $|P'| = |P|$.*

Proof. Since P' is a path of H, and P is a longest path of H, it holds clearly that $|P'| \leq |P|$. Consider now a longest path P of H, and let k and ℓ be the number of connector and stable vertices of P, respectively. Since no two stable vertices of H are adjacent due to Observation 2, it holds clearly that $\ell \leq k + 1$. Similarly to the second part of the proof of Lemma 12, we can obtain a path $\widehat{P}$ of H with k vertices, by removing all ℓ stable vertices from P. Then, similarly to the first part of the proof of Lemma 12, there exists a binormal path P' of H, where $|P'| = 2k + 1 \geq k + \ell = |P|$. It follows that $|P'| = |P|$, for any longest path P and any longest binormal path P' of H. □

Let P be the longest binormal path of H computed in Step 2 of Algorithm LP_Interval, using Algorithm LP_on_H. Then, in Step 3 Algorithm LP_Interval computes the path $\widehat{P}$ by deleting all stable vertices from P. By the construction of H, all neighbors of a stable vertex a are connector vertices and form a clique in G; thus, for every subpath (v, a, v') of P, v is adjacent to v' in G. It follows that $\widehat{P}$ is a path of G. Moreover, since P is binormal, it has k connector vertices and $k + 1$ stable vertices, i.e., $|P| = 2k + 1$, where $k \geq 1$. Thus, since we have removed all $k + 1$ stable vertices of P, it follows that $|\widehat{P}| = k$ and, thus, $\widehat{P}$ is a longest path of G due to Lemma 12. Therefore, we have proved the following result.

Theorem 1. *Algorithm LP_Interval computes a longest path of an interval graph G.*

4.3 Time Complexity

Let G be an interval graph on $|V(G)| = n$ vertices and $|E(G)| = m$ edges. It has been shown that we can obtain the right-end ordering π of G, which results from numbering the intervals after sorting them on their right ends, in $O(n + m)$ time [1, 20].

First, we show that Step 1 of Algorithm LP_Interval, which constructs the stable-connection graph H of the graph G, takes $O(n^2)$ time. Indeed, for every connector vertex u_i, $1 \leq i \leq n$, we can add two stable vertices in $V(H)$ in $O(1)$ time and we can compute the specific neighborhood of u_i in $O(n)$ time.

Step 2 of Algorithm LP_Interval includes the execution of Algorithm LP_on_H. The subroutine `process()` takes $O(n^2)$ time, due to the $O(n^2)$ pairs of the neighbors u_x and u_y of the connector vertex u_j in the graph $H(i, j)$. Additionally, the subroutine `process()` is executed at most once for each subgraph $H(i, j)$ of H, $1 \leq i \leq j \leq n$, i.e., it is executed $O(n^2)$ times. Thus, Algorithm LP_on_H takes $O(n^4)$ time.

Step 3 of Algorithm LP_Interval can be executed in $O(n)$ time since we simply traverse the vertices of the path P, constructed by Algorithm LP_on_H, and delete every stable vertex.

Therefore, we obtain the following result concerning the time complexity of the algorithm.

Theorem 2. *A longest path of an interval graph can be computed in $O(n^4)$ time.*

In order to compute the length of a longest path, we need to store one value for every induced subgraph $H(i,j)$ and for every stable vertex u_y of $H(i,j)$. Thus, since there are in total $O(n^2)$ such subgraphs $H(i,j)$, $1 \leq i \leq j \leq n$, and since each one has at most $O(n)$ stable vertices, we can compute the length of a longest path in $O(n^3)$ space. Furthermore, in order to compute and report a longest path, instead of its length only, we have to store a path of at most n vertices for every one of the $O(n^3)$ computed values. Therefore, the space complexity of Algorithm LP_Interval is $O(n^4)$.

5 Concluding Remarks

In this paper we presented a polynomial-time algorithm for solving the longest path problem on interval graphs, which runs in $O(n^4)$ time and, thus, provided a solution to the open problem stated by Uehara and Uno in [22] asking for the complexity status of the longest path problem on interval graphs. It would be interesting to see whether the ideas presented in this paper can be applied to find a polynomial solution to the longest path problem on convex and biconvex graphs, the complexities of which still remain open [22].

References

1. S.R. Arikati and C. Pandu Rangan, Linear algorithm for optimal path cover problem on interval graphs, *Inform. Proc. Lett.* **35** (1990) 149–153.
2. A.A. Bertossi, Finding Hamiltonian circuits in proper interval graphs, *Inform. Proc. Lett.* **17** (1983) 97–101.
3. A. Björklund and T. Husfeldt, Finding a path of superlogarithmic length, *SIAM J. Computing* **32** (2003) 1395–1402.
4. R. Bulterman, F. van der Sommen, G. Zwaan, T. Verhoeff, A. van Gasteren, and W. Feijen, On computing a longest path in a tree, *Inform. Proc. Lett.* **81** (2002) 93–96.
5. P. Damaschke, J.S. Deogun, D. Kratsch, and G. Steiner, Finding Hamiltonian paths in cocomparability graphs using the bump number algorithm, *Order* **8** (1992) 383–391.
6. P. Damaschke, The Hamiltonian circuit problem for circle graphs is NP-complete, *Inform. Proc. Lett.* **32** (1989) 1–2.
7. P. Damaschke, Paths in interval graphs and circular arc graphs. *Discrete Math.* **112** (1993) 49–64.
8. T. Feder and R. Motwani, Finding large cycles in Hamiltonian graphs, *Proc. 16th annual ACM-SIAM Symp. on Discrete Algorithms (SODA)*, ACM (2005) 166–175.
9. H.N. Gabow, Finding paths and cycles of superpolylogarithmic length, *Proc. 36th annual ACM Symp. on Theory of Computing (STOC)*, ACM (2004) 407–416.
10. H.N. Gabow and S. Nie, Finding long paths, cycles and circuits, *19th annual International Symp. on Algorithms and Computation (ISAAC)*, LNCS **5369** (2008) 752–763.
11. M.R. Garey and D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-completeness*, W.H. Freeman, San Francisco, 1979.
12. M.R. Garey, D.S. Johnson, and R.E. Tarjan, The planar Hamiltonian circuit problem is NP-complete, *SIAM J. Computing* **5** (1976) 704–714.
13. P.W. Goldberg, M.C. Golumbic, H. Kaplan, and R. Shamir, Four strikes against physical mapping of DNA, *Journal of Computational Biology* **2** (1995) 139–152.
14. M.C. Golumbic, *Algorithmic Graph Theory and Perfect Graphs* (Annals of Discrete Mathematics, Vol. 57), North-Holland Publishing Co., Amsterdam, The Netherlands, 2004.

15. A. Itai, C.H. Papadimitriou, and J.L. Szwarcfiter, Hamiltonian paths in grid graphs, *SIAM J. Computing* **11** (1982) 676–686.
16. D. Karger, R. Motwani, and G.D.S. Ramkumar, On approximating the longest path in a graph, *Algorithmica* **18** (1997) 82–98.
17. J.M. Keil, Finding Hamiltonian circuits in interval graphs, *Inform. Proc. Lett.* **20** (1985) 201–206.
18. G.K. Manacher, T.A. Mankus, and C.J. Smith, An optimum $\Theta(nlogn)$ algorithm for finding a canonical Hamiltonian path and a canonical Hamiltonian circuit in a set of intervals, *Inform. Proc. Lett.* **35** (1990) 205–211.
19. H. Müller, Hamiltonian circuits in chordal bipartite graphs, *Discrete Math.* **156** (1996) 291–298.
20. G. Ramalingam and C. Pandu Rangan, A unified approach to domination problems on interval graphs, *Inform. Proc. Lett.* **27** (1988) 271–274.
21. Y. Takahara, S. Teramoto, and R. Uehara, Longest path problems on ptolemaic graphs, *IEICE Trans. Inf. and Syst.* **91-D** (2008) 170–177.
22. R. Uehara and Y. Uno, Efficient algorithms for the longest path problem, *15th annual International Symp. on Algorithms and Computation (ISAAC)*, LNCS **3341** (2004) 871–883.
23. R. Uehara and G. Valiente, Linear structure of bipartite permutation graphs and the longest path problem, *Inform. Proc. Lett.* **103** (2007) 71–77.
24. S. Vishwanathan, An approximation algorithm for finding a long path in Hamiltonian graphs, *Proc. 11th annual ACM-SIAM Symp. on Discrete Algorithms (SODA)*, ACM (2000) 680–685.
25. Z. Zhang, and H. Li, Algorithms for long paths in graphs, *Theoret. Comput. Sci.* **377** (2007) 25–34.

Aachener Informatik-Berichte

This list contains all technical reports published during the past five years. A complete list of reports dating back to 1987 is available from http://aib.informatik.rwth-aachen.de/. To obtain copies consult the above URL or send your request to: Informatik-Bibliothek, RWTH Aachen, Ahornstr. 55, 52056 Aachen, Email: `biblio@informatik.rwth-aachen.de`

2004-01 * Fachgruppe Informatik: Jahresbericht 2003

2004-02 Benedikt Bollig, Martin Leucker: Message-Passing Automata are expressively equivalent to EMSO logic

2004-03 Delia Kesner, Femke van Raamsdonk, Joe Wells (eds.): HOR 2004 – 2nd International Workshop on Higher-Order Rewriting

2004-04 Slim Abdennadher, Christophe Ringeissen (eds.): RULE 04 – Fifth International Workshop on Rule-Based Programming

2004-05 Herbert Kuchen (ed.): WFLP 04 – 13th International Workshop on Functional and (Constraint) Logic Programming

2004-06 Sergio Antoy, Yoshihito Toyama (eds.): WRS 04 – 4th International Workshop on Reduction Strategies in Rewriting and Programming

2004-07 Michael Codish, Aart Middeldorp (eds.): WST 04 – 7th International Workshop on Termination

2004-08 Klaus Indermark, Thomas Noll: Algebraic Correctness Proofs for Compiling Recursive Function Definitions with Strictness Information

2004-09 Joachim Kneis, Daniel Mölle, Stefan Richter, Peter Rossmanith: Parameterized Power Domination Complexity

2004-10 Zinaida Benenson, Felix C. Gärtner, Dogan Kesdogan: Secure Multi-Party Computation with Security Modules

2005-01 * Fachgruppe Informatik: Jahresbericht 2004

2005-02 Maximillian Dornseif, Felix C. Gärtner, Thorsten Holz, Martin Mink: An Offensive Approach to Teaching Information Security: "Aachen Summer School Applied IT Security"

2005-03 Jürgen Giesl, René Thiemann, Peter Schneider-Kamp: Proving and Disproving Termination of Higher-Order Functions

2005-04 Daniel Mölle, Stefan Richter, Peter Rossmanith: A Faster Algorithm for the Steiner Tree Problem

2005-05 Fabien Pouget, Thorsten Holz: A Pointillist Approach for Comparing Honeypots

2005-06 Simon Fischer, Berthold Vöcking: Adaptive Routing with Stale Information

2005-07 Felix C. Freiling, Thorsten Holz, Georg Wicherski: Botnet Tracking: Exploring a Root-Cause Methodology to Prevent Distributed Denial-of-Service Attacks

2005-08 Joachim Kneis, Peter Rossmanith: A New Satisfiability Algorithm With Applications To Max-Cut

2005-09 Klaus Kursawe, Felix C. Freiling: Byzantine Fault Tolerance on General Hybrid Adversary Structures

2005-10 Benedikt Bollig: Automata and Logics for Message Sequence Charts

2005-11 Simon Fischer, Berthold Vöcking: A Counterexample to the Fully Mixed Nash Equilibrium Conjecture

2005-12 Neeraj Mittal, Felix Freiling, S. Venkatesan, Lucia Draque Penso: Efficient Reductions for Wait-Free Termination Detection in Faulty Distributed Systems

2005-13 Carole Delporte-Gallet, Hugues Fauconnier, Felix C. Freiling: Revisiting Failure Detection and Consensus in Omission Failure Environments

2005-14 Felix C. Freiling, Sukumar Ghosh: Code Stabilization

2005-15 Uwe Naumann: The Complexity of Derivative Computation

2005-16 Uwe Naumann: Syntax-Directed Derivative Code (Part I: Tangent-Linear Code)

2005-17 Uwe Naumann: Syntax-directed Derivative Code (Part II: Intraprocedural Adjoint Code)

2005-18 Thomas von der Maßen, Klaus Müller, John MacGregor, Eva Geisberger, Jörg Dörr, Frank Houdek, Harbhajan Singh, Holger Wußmann, Hans-Veit Bacher, Barbara Paech: Einsatz von Features im Software-Entwicklungsprozess - Abschlußbericht des GI-Arbeitskreises “Features”

2005-19 Uwe Naumann, Andre Vehreschild: Tangent-Linear Code by Augmented LL-Parsers

2005-20 Felix C. Freiling, Martin Mink: Bericht über den Workshop zur Ausbildung im Bereich IT-Sicherheit Hochschulausbildung, berufliche Weiterbildung, Zertifizierung von Ausbildungsangeboten am 11. und 12. August 2005 in Köln organisiert von RWTH Aachen in Kooperation mit BITKOM, BSI, DLR und Gesellschaft fuer Informatik (GI) e.V.

2005-21 Thomas Noll, Stefan Rieger: Optimization of Straight-Line Code Revisited

2005-22 Felix Freiling, Maurice Herlihy, Lucia Draque Penso: Optimal Randomized Fair Exchange with Secret Shared Coins

2005-23 Heiner Ackermann, Alantha Newman, Heiko Röglin, Berthold Vöcking: Decision Making Based on Approximate and Smoothed Pareto Curves

2005-24 Alexander Becher, Zinaida Benenson, Maximillian Dornseif: Tampering with Motes: Real-World Physical Attacks on Wireless Sensor Networks

2006-01 * Fachgruppe Informatik: Jahresbericht 2005

2006-02 Michael Weber: Parallel Algorithms for Verification of Large Systems

2006-03 Michael Maier, Uwe Naumann: Intraprocedural Adjoint Code Generated by the Differentiation-Enabled NAGWare Fortran Compiler

2006-04 Ebadollah Varnik, Uwe Naumann, Andrew Lyons: Toward Low Static Memory Jacobian Accumulation

2006-05 Uwe Naumann, Jean Utke, Patrick Heimbach, Chris Hill, Derya Ozyurt, Carl Wunsch, Mike Fagan, Nathan Tallent, Michelle Strout: Adjoint Code by Source Transformation with OpenAD/F

2006-06 Joachim Kneis, Daniel Mölle, Stefan Richter, Peter Rossmanith: Divide-and-Color

2006-07 Thomas Colcombet, Christof Löding: Transforming structures by set interpretations

2006-08 Uwe Naumann, Yuxiao Hu: Optimal Vertex Elimination in Single-Expression-Use Graphs

2006-09 Tingting Han, Joost-Pieter Katoen: Counterexamples in Probabilistic Model Checking

2006-10 Mesut Günes, Alexander Zimmermann, Martin Wenig, Jan Ritzerfeld, Ulrich Meis: From Simulations to Testbeds - Architecture of the Hybrid MCG-Mesh Testbed

2006-11 Bastian Schlich, Michael Rohrbach, Michael Weber, Stefan Kowalewski: Model Checking Software for Microcontrollers

2006-12 Benedikt Bollig, Joost-Pieter Katoen, Carsten Kern, Martin Leucker: Replaying Play in and Play out: Synthesis of Design Models from Scenarios by Learning

2006-13 Wong Karianto, Christof Löding: Unranked Tree Automata with Sibling Equalities and Disequalities

2006-14 Danilo Beuche, Andreas Birk, Heinrich Dreier, Andreas Fleischmann, Heidi Galle, Gerald Heller, Dirk Janzen, Isabel John, Ramin Tavakoli Kolagari, Thomas von der Maßen, Andreas Wolfram: Report of the GI Work Group “Requirements Management Tools for Product Line Engineering”

2006-15 Sebastian Ullrich, Jakob T. Valvoda, Torsten Kuhlen: Utilizing optical sensors from mice for new input devices

2006-16 Rafael Ballagas, Jan Borchers: Selexels: a Conceptual Framework for Pointing Devices with Low Expressiveness

2006-17 Eric Lee, Henning Kiel, Jan Borchers: Scrolling Through Time: Improving Interfaces for Searching and Navigating Continuous Audio Timelines

2007-01 * Fachgruppe Informatik: Jahresbericht 2006

2007-02 Carsten Fuhs, Jürgen Giesl, Aart Middeldorp, Peter Schneider-Kamp, René Thiemann, and Harald Zankl: SAT Solving for Termination Analysis with Polynomial Interpretations

2007-03 Jürgen Giesl, René Thiemann, Stephan Swiderski, and Peter Schneider-Kamp: Proving Termination by Bounded Increase

2007-04 Jan Buchholz, Eric Lee, Jonathan Klein, and Jan Borchers: coJIVE: A System to Support Collaborative Jazz Improvisation

2007-05 Uwe Naumann: On Optimal DAG Reversal

2007-06 Joost-Pieter Katoen, Thomas Noll, and Stefan Rieger: Verifying Concurrent List-Manipulating Programs by LTL Model Checking

2007-07 Alexander Nyßen, Horst Lichter: MeDUSA - MethoD for UML2-based Design of Embedded Software Applications

2007-08 Falk Salewski and Stefan Kowalewski: Achieving Highly Reliable Embedded Software: An empirical evaluation of different approaches

2007-09 Tina Kraußer, Heiko Mantel, and Henning Sudbrock: A Probabilistic Justification of the Combining Calculus under the Uniform Scheduler Assumption

2007-10 Martin Neuhäußer, Joost-Pieter Katoen: Bisimulation and Logical Preservation for Continuous-Time Markov Decision Processes

2007-11 Klaus Wehrle (editor): 6. Fachgespräch Sensornetzwerke

2007-12 Uwe Naumann: An L-Attributed Grammar for Adjoint Code

2007-13 Uwe Naumann, Michael Maier, Jan Riehme, and Bruce Christianson: Second-Order Adjoints by Source Code Manipulation of Numerical Programs

2007-14 Jean Utke, Uwe Naumann, Mike Fagan, Nathan Tallent, Michelle Strout, Patrick Heimbach, Chris Hill, and Carl Wunsch: OpenAD/F: A Modular, Open-Source Tool for Automatic Differentiation of Fortran Codes

2007-15 Volker Stolz: Temporal assertions for sequential and concurrent programs

2007-16 Sadeq Ali Makram, Mesut Güneç, Martin Wenig, Alexander Zimmermann: Adaptive Channel Assignment to Support QoS and Load Balancing for Wireless Mesh Networks

2007-17 René Thiemann: The DP Framework for Proving Termination of Term Rewriting

2007-18 Uwe Naumann: Call Tree Reversal is NP-Complete

2007-19 Jan Riehme, Andrea Walther, Jörg Stiller, Uwe Naumann: Adjoints for Time-Dependent Optimal Control

2007-20 Joost-Pieter Katoen, Daniel Klink, Martin Leucker, and Verena Wolf: Three-Valued Abstraction for Probabilistic Systems

2007-21 Tingting Han, Joost-Pieter Katoen, and Alexandru Mereacre: Compositional Modeling and Minimization of Time-Inhomogeneous Markov Chains

2007-22 Heiner Ackermann, Paul W. Goldberg, Vahab S. Mirrokni, Heiko Röglin, and Berthold Vöcking: Uncoordinated Two-Sided Markets

2008-01 * Fachgruppe Informatik: Jahresbericht 2007

2008-02 Henrik Bohnenkamp, Marielle Stoelinga: Quantitative Testing

2008-03 Carsten Fuhs, Jürgen Giesl, Aart Middeldorp, Peter Schneider-Kamp, René Thiemann, Harald Zankl: Maximal Termination

2008-04 Uwe Naumann, Jan Riehme: Sensitivity Analysis in Sisyphe with the AD-Enabled NAGWare Fortran Compiler

2008-05 Frank G. Radmacher: An Automata Theoretic Approach to the Theory of Rational Tree Relations

2008-06 Uwe Naumann, Laurent Hascoet, Chris Hill, Paul Hovland, Jan Riehme, Jean Utke: A Framework for Proving Correctness of Adjoint Message Passing Programs

2008-07 Alexander Nyßen, Horst Lichter: The MeDUSA Reference Manual, Second Edition

2008-08 George B. Mertzios, Stavros D. Nikolopoulos: The λ-cluster Problem on Parameterized Interval Graphs

2008-09 George B. Mertzios, Walter Unger: An optimal algorithm for the k-fixed-endpoint path cover on proper interval graphs

2008-10 George B. Mertzios, Walter Unger: Preemptive Scheduling of Equal-Length Jobs in Polynomial Time

2008-11 George B. Mertzios: Fast Convergence of Routing Games with Splittable Flows

2008-12 Joost-Pieter Katoen, Daniel Klink, Martin Leucker, Verena Wolf: Abstraction for stochastic systems by Erlang's method of stages

2008-13 Beatriz Alarcón, Fabian Emmes, Carsten Fuhs, Jürgen Giesl, Raúl Gutiérrez, Salvador Lucas, Peter Schneider-Kamp, René Thiemann: Improving Context-Sensitive Dependency Pairs

2008-14 Bastian Schlich: Model Checking of Software for Microcontrollers

2008-15 Joachim Kneis, Alexander Langer, Peter Rossmanith: A New Algorithm for Finding Trees with Many Leaves

2008-16 Hendrik vom Lehn, Elias Weingärtner and Klaus Wehrle: Comparing recent network simulators: A performance evaluation study

2008-17 Peter Schneider-Kamp: Static Termination Analysis for Prolog using Term Rewriting and SAT Solving

2008-18 Falk Salewski: Empirical Evaluations of Safety-Critical Embedded Systems

2009-03 Alexander Nyßen: Model-Based Construction of Embedded Real-Time Software - A Methodology for Small Devices

2009-05 George B. Mertzios, Ignasi Sau, Shmuel Zaks: A New Intersection Model and Improved Algorithms for Tolerance Graphs

2009-06 George B. Mertzios, Ignasi Sau, Shmuel Zaks: The Recognition of Tolerance and Bounded Tolerance Graphs is NP-complete

2009-07 Joachim Kneis, Alexander Langer, Peter Rossmanith: Derandomizing Non-uniform Color-Coding I

2009-08 Joachim Kneis, Alexander Langer: Satellites and Mirrors for Solving Independent Set on Sparse Graphs

* These reports are only available as a printed version.

Please contact `biblio@informatik.rwth-aachen.de` to obtain copies.