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 University, 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 [20], 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 numerous 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 [15]. Moreover, there is no polynomial-time constant-factor approximation algorithm for the longest path problem unless P=NP [15]. For related results see also [7–9, 22, 23].

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

⋆ This research is co-financed by E.U.-European Social Fund (80%) and the Greek Ministry of Development-GSRT (20%).
⋆⋆ This research is partially supported by Empirikion Foundation, Greece.

problem is known to be NP-complete in general graphs [10, 11], and remains NP-complete even when restricted to some small classes of graphs such as split graphs [13], chordal bipartite graphs, split strongly chordal graphs [17], circle graphs [5], planar graphs [11], and grid graphs [14]. However, it makes sense to investigate the tractability of the longest path problem on the classes of graphs for which the Hamiltonian path problem admits polynomial time solutions. Such classes include interval graphs [16], circular-arc graphs [6], convex bipartite graphs [17], and co-comparability graphs [4]. Note that the problem of finding a longest path on proper interval graphs is easy, since all connected proper interval graphs have a Hamiltonian path which can be computed in linear time [2]. On the contrary, not all interval graphs are Hamiltonian; in the case where an interval graph has a Hamiltonian path, it can be computed in linear time [16]. However, in the case where an interval graph is not Hamiltonian, there is no known algorithm for finding a longest path on it.

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 [3]. Later, through a generalization of Dijkstra's algorithm for trees, Uehara and Uno [20] 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 [21], and on ptolemaic graphs in $O(n^5)$ time and $O(n^2)$ space [19].

Furthermore, Uehara and Uno in [20] 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 [20] 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 [13]; 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 [20], but also improve the known time complexity of the problem on interval biconvex graphs, a subclass of interval graphs [20].

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, 13, 16]). Moreover, interval graphs

have received a lot of attention due to their applicability to DNA physical mapping problems [12], and find many applications in several fields and disciplines such as genetics, molecular biology, scheduling, VLSI circuit design, archaeology and psychology [13].

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 undirected edge is a pair of distinct vertices $u, v \in V(G)$, and is denoted by uv. We say that the vertex u is adjacent to the vertex v or, equivalently, the vertex u sees the vertex v, if there is an edge uv in G. 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 $v \in V(G)$.

A *simple path* of a graph G is a sequence of distinct vertices $v_1, v_2, \ldots, v_k$ such that $v_i v_{i+1} \in E(G)$, for each i, $1 \leq i \leq k-1$, and is denoted by $(v_1, v_2, \ldots, v_k)$; throughout the paper all paths considered are simple. 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. Moreover, 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 a graph. Sometimes, we shall denote the path P by $P = (v_1, v_2, \ldots, v_{i-1}, P_0, v_{j+1}, v_{j+2}, \ldots, v_k)$.

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 [18] proposed a numbering of the vertices of an interval graph; they stated the following lemma.

Lemma 1. *(Ramalingam and Rangan [18]): 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 [18], 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, 18]. 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 $i < j$.

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, $2 \leq i \leq k$, such that $v_{i-1}v_\ell \in E(G)$ and $v_\ell <_\pi v_i$.*

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.

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 $V(P') = V(P)$.*

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 $V(H)$ and make both these vertices to be adjacent to every vertex in $N_G[v_i] \cap \{v_i, v_{i+1}, \ldots, v_{|V(G)|}\}$; add $a_{i,1}$ and $a_{i,2}$ to A. 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.*

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:*

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), ∀ u_k ∈ A(H(f(u_j) + 1, j − 1));
```

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

(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.

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.*

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.

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 $P(u_k; i, j)$.*

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.

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.*

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 $H(i,j)$.*

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.*

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 (sketch). 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 indeed a binormal path with u_y as its right endpoint. □

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 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 the graph 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. □

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

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. Thus, 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, 18].

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.

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

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 [20] 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 [20].

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. 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.
4. 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.
5. P. Damaschke, The Hamiltonian circuit problem for circle graphs is NP-complete, *Inform. Proc. Lett.* **32** (1989) 1–2.

6. P. Damaschke, Paths in interval graphs and circular arc graphs. *Discrete Math.* **112** (1993) 49–64.
7. 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.
8. H.N. Gabow, Finding paths and cycles of superpolylogarithmic length, *Proc. 36th annual ACM Symp. on Theory of Computing (STOC)*, ACM (2004) 407–416.
9. 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.
10. M.R. Garey and D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-completeness*, W.H. Freeman, San Francisco, 1979.
11. 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.
12. 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.
13. M.C. Golumbic, *Algorithmic Graph Theory and Perfect Graphs* (Annals of Discrete Mathematics, Vol. 57), North-Holland Publishing Co., Amsterdam, The Netherlands, 2004.
14. A. Itai, C.H. Papadimitriou, and J.L. Szwarcfiter, Hamiltonian paths in grid graphs, *SIAM J. Computing* **11** (1982) 676–686.
15. D. Karger, R. Motwani, and G.D.S. Ramkumar, On approximating the longest path in a graph, *Algorithmica* **18** (1997) 82–98.
16. J.M. Keil, Finding Hamiltonian circuits in interval graphs, *Inform. Proc. Lett.* **20** (1985) 201–206.
17. H. Müller, Hamiltonian circuits in chordal bipartite graphs, *Discrete Math.* **156** (1996) 291–298.
18. G. Ramalingam and C. Pandu Rangan, A unified approach to domination problems on interval graphs, *Inform. Proc. Lett.* **27** (1988) 271–274.
19. Y. Takahara, S. Teramoto, and R. Uehara, Longest path problems on ptolemaic graphs, *IEICE Trans. Inf. and Syst.* **91-D** (2008) 170–177.
20. 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.
21. R. Uehara and G. Valiente, Linear structure of bipartite permutation graphs and the longest path problem, *Inform. Proc. Lett.* **103** (2007) 71–77.
22. 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.
23. Z. Zhang, and H. Li, Algorithms for long paths in graphs, *Theoret. Comput. Sci.* **377** (2007) 25–34.