Algorithmica
DOI 10.1007/s00453-009-9292-5

The 1-Fixed-Endpoint Path Cover Problem is Polynomial on Interval Graphs

Katerina Asdre · Stavros D. Nikolopoulos

Received: 1 April 2008 / Accepted: 11 February 2009

Abstract We consider a variant of the path cover problem, namely, the k-fixed-endpoint path cover problem, or kPC for short, on interval graphs. Given a graph G and a subset $\mathcal{T}$ of k vertices of $V(G)$, a k-fixed-endpoint path cover of G with respect to $\mathcal{T}$ is a set of vertex-disjoint paths $\mathcal{P}$ that covers the vertices of G such that the k vertices of $\mathcal{T}$ are all endpoints of the paths in $\mathcal{P}$. The kPC problem is to find a k-fixed-endpoint path cover of G of minimum cardinality; note that, if $\mathcal{T}$ is empty the stated problem coincides with the classical path cover problem. In this paper, we study the 1-fixed-endpoint path cover problem on interval graphs, or 1PC for short, generalizing the 1HP problem which has been proved to be NP-complete even for small classes of graphs. Motivated by a work of Damaschke (Discrete Math. 112:49–64, 1993), where he left both 1HP and 2HP problems open for the class of interval graphs, we show that the 1PC problem can be solved in polynomial time on the class of interval graphs. We propose a polynomial-time algorithm for the problem, which also enables us to solve the 1HP problem on interval graphs within the same time and space complexity.

Keywords Perfect graphs · Interval graphs · Path cover · Fixed-endpoint path cover · Linear-time algorithms

1 Introduction

Framework—Motivation A well studied problem with numerous practical applications in graph theory is to find a minimum number of vertex-disjoint paths of a

K. Asdre · S.D. Nikolopoulos (✉)
Department of Computer Science, University of Ioannina, P.O. Box 1186, 45110 Ioannina, Greece
e-mail: stavros@cs.uoi.gr

K. Asdre
e-mail: katerina@cs.uoi.gr

Published online: 04 March 2009

graph G that cover the vertices of G. This problem, also known as the path cover problem (PC), finds application in the fields of database design, networks, code optimization among many others (see [1, 2, 19, 28]); it is well known that the path cover problem and many of its variants are NP-complete in general graphs [10]. A graph that admits a path cover of size one is referred to as Hamiltonian. Thus, the path cover problem is at least as hard as the Hamiltonian path problem (HP), that is, the problem of deciding whether a graph is Hamiltonian. The path cover problem is known to be NP-complete even when the input is restricted to several interesting special classes of graphs; for example, it is NP-complete on planar graphs [11], bipartite graphs [12], chordal graphs [12], chordal bipartite graphs [20] and strongly chordal graphs [20]. Bertossi and Bonuccelli [6] proved that the Hamiltonian Circuit problem is NP-complete on several interesting classes of intersection graphs.

Several variants of the HP problem are also of great interest, among which is the problem of deciding whether a graph admits a Hamiltonian path between two points (2HP). The 2HP problem is the same as the HP problem except that in 2HP two vertices of the input graph G are specified, say, u and v, and we are asked whether G contains a Hamiltonian path beginning with u and ending with v. Similarly, the 1HP problem is to determine whether a graph G admits a Hamiltonian path starting from a specific vertex u of G, and to find one if such a path does exist. Both 1HP and 2HP problems are also NP-complete in general graphs [10].

The path cover problem as well as several variants of it have been extensively studied due to their wide applicability in many fields. Some of these problems, of both theoretical and practical importance, are in the context of communication and/or transposition networks [29]. In such problems, we are given a graph (network) G and two disjoint subsets $\mathcal{T}_1$ and $\mathcal{T}_2$ of vertices of G, and the objective is to determine whether G admits λ vertex-disjoint paths with several conditions on their endpoints with respect to $\mathcal{T}_1$ and $\mathcal{T}_2$, e.g., paths with both their endpoints in $\mathcal{T}_1 \cup \mathcal{T}_2$, paths with one endpoint in $\mathcal{T}_1$ and the other in $\mathcal{T}_2$, etc. [3, 4, 29]; note that, the endpoints of a path P are the first vertex and the last vertex visited by P.

A similar problem that has received increased attention in recent years is in the context of communication networks. The only efficient way to transmit high volume communication, such as in multimedia applications, is through disjoint paths that are dedicated to pairs of processors. To efficiently utilize the network, one needs a simple algorithm that, with minimum overhead, constructs a large number of edge-disjoint paths between pairs of two given sets $\mathcal{T}_1$ and $\mathcal{T}_2$ of requests.

Furthermore, in the study of interconnection networks, the reliability of the interconnection network subject to node failures corresponds to the connectivity of an interconnection graph. It is well-known that the connectivity of a graph G is characterized in terms of vertex-disjoint paths joining a pair of vertices in G. Thus, one-to-many vertex-disjoint paths joining a vertex s (source) and k distinct vertices $t_1, t_2, \dots, t_k$ (sinks) are required. A related work was presented by Park in [23].

Another related problem is the disjoint paths (DP) problem, which is defined as follows: Given a graph G and pairs $(s_1, t_1), (s_2, t_2), \dots, (s_k, t_k)$ of vertices of G, the objective is to determine whether G admits k vertex-disjoint paths $P_1, P_2, \dots, P_k$ in G such that P_i joins s_i and t_i $(1 \le i \le k)$. The problem was shown to be NP-complete by Karp [17] if k is a variable part of the input. For fixed k, however, the problem is more tractable; a polynomial-time algorithm was described by Robertson and Seymour [25]. Note that, for $k=2$, there are several polynomial-time algorithms in the literature for the DP problem [26, 27, 30]. In contrast, the corresponding question for directed graphs G where we seek directed paths $P_1, P_2, \dots, P_k$ is NP-complete even for $k=2$ [9].

In [8], Damaschke provided a foundation for obtaining polynomial-time algorithms for several problems concerning paths in interval graphs, such as finding Hamiltonian paths and circuits, and partitions into paths. In the same paper, he stated that the complexity status of both 1HP and 2HP problems on interval graphs remains an open question.

Motivated by the above issues, we state a variant of the path cover problem, namely, the 1-fixed-endpoint path cover (1PC) problem, which generalizes the 1HP problem.

Problem 1PC Given a graph G and a vertex $u \in V(G)$, a *1-fixed-endpoint path cover* of the graph G with respect to u is a path cover of G such that the vertex u is an endpoint of a path in the path cover; a *minimum* 1*-fixed-endpoint path cover* of G with respect to u is a 1-fixed-endpoint path cover of G with minimum cardinality; the 1*-fixed-endpoint path cover problem* (1PC) is to find a minimum 1-fixed-endpoint path cover of the graph G.

Contribution In this paper, we study the complexity status of the 1-fixed-endpoint path cover problem (1PC) on the class of interval graphs [7, 12], and show that this problem can be solved in polynomial time. The proposed algorithm runs in $O(n^3)$ time on an interval graph G on n vertices and m edges and requires linear space. The proposed algorithm for the 1PC problem can also be used to solve the 1HP problem on interval graphs within the same time and space complexity. Using our algorithm for the 1PC problem and a simple reduction described by Müller in [20], we solve the HP problem on a X-convex graph $G(X, Y, E)$ with $|Y| - |X| = 1$, which was left open in [31]. We also show that the 1HP problem on a biconvex graph G is solvable in $O(n^3)$ time. Figure 1 shows a diagram of class inclusions for a number of graph classes, subclasses of comparability and chordal graphs, and the current complexity status of the 1HP problem on these classes; for definitions of the classes shown, see [7, 12].

Related Work Interval graphs form an important class of perfect graphs [12] and many problems that are NP-complete on arbitrary graphs are shown to admit polynomial time algorithms on this class [2, 12, 18]. Both Hamiltonian Circuit (HC) and Hamiltonian Path (HP) problems are polynomially solvable for the class of interval and proper interval graphs. Keil introduced a linear-time algorithm for the HC problem on interval graphs [18] and Arikati and Rangan [2] presented a linear-time algorithm for the minimum path cover problem on interval graphs. Bertossi [5] proved that a proper interval graph has a Hamiltonian path if and only if it is connected. He also gave an $O(n \log n)$ algorithm for finding a Hamiltonian circuit in a proper interval graph. Recently, Asdre and Nikolopoulos proposed a linear-time algorithm for the k-fixed-endpoint path cover problem (kPC) on cographs and on proper interval

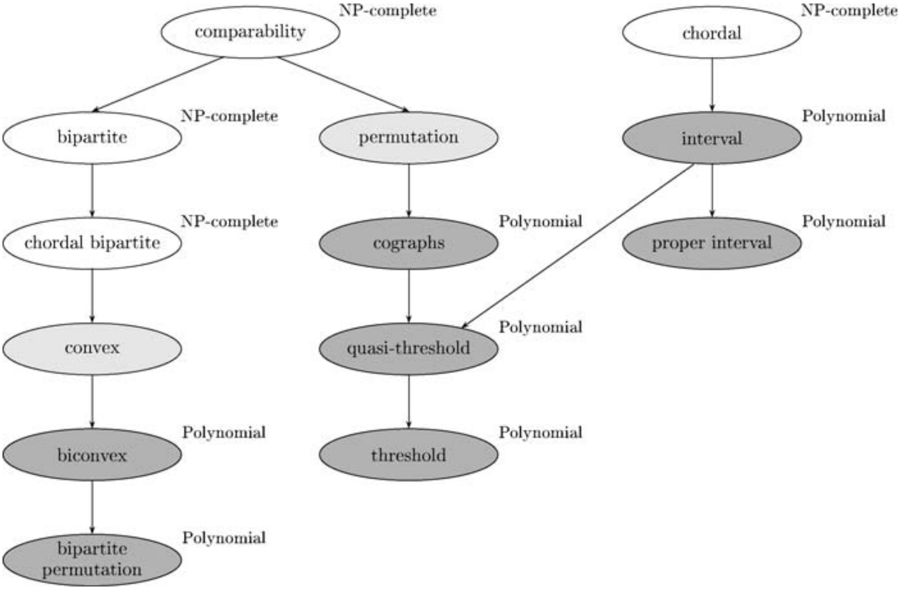


Fig. 1 The complexity status (NP-complete, unknown, polynomial) of the 1HP problem for some graph subclasses of comparability and chordal graphs. $A \to B$ indicates that class A contains class B

graphs [3, 4]. Furthermore, Lin et al. [19] proposed an optimal algorithm for the path cover problem on cographs while Nakano et al. [21] proposed an optimal parallel algorithm which finds and reports all the paths in a minimum path cover of a cograph in $O(\log n)$ time using $O(n/\log n)$ processors on a PRAM model. Hsieh et al. [14] presented an $O(n+m)$-time sequential algorithm for the Hamiltonian problem on a distance-hereditary graph and also proposed a parallel implementation of their algorithm which solves the problem in $O(\log n)$ time using $O((n+m)/\log n)$ processors on a PRAM model. A unified approach to solving the Hamiltonian problems on distance-hereditary graphs was presented in [15], while Hsieh [13] presented an efficient parallel strategy for the 2HP problem on the same class of graphs. Algorithms for the path cover problem on other classes of graphs were proposed in [16, 22, 28].

Road Map The paper is organized as follows. In Sect. 2 we establish the notation and related terminology, and we present background results. In Sect. 3 we describe our algorithm for the 1PC problem, while in Sect. 4 we prove its correctness and compute its time and space complexity. Section 5 presents some related results and in Sect. 6 we conclude the paper and discuss possible future extensions.

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. Let S be a subset of the vertex set of a graph G. Then, the subgraph of G induced by S is denoted by $G[S]$. Furthermore, we say that a vertex v *sees* vertex v' if and only if $vv' \in E(G)$.

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 their corresponding intervals intersect. F is called an *intersection model* for G [2]. Interval graphs find applications in genetics, molecular biology, archeology, and storage information retrieval [12]. Interval graphs form an important class of perfect graphs [12] and many problems that are NP-complete on arbitrary graphs are shown to admit polynomial time algorithms on this class [2, 12, 18]. The class of interval graphs is *hereditary*, that is, every induced subgraph of an interval graph G is also an interval graph. We state the following numbering for the vertices of an interval graph proposed in [24].

Lemma 2.1 (Ramalingam and Rangan [24]) *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 $jk \in E(G)$.*

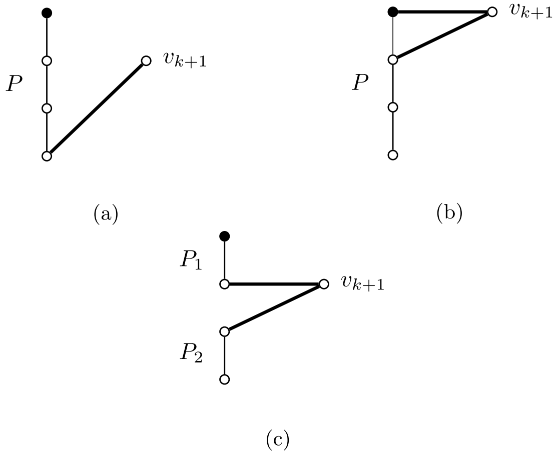
As shown in [24], the numbering of Lemma 2.1, which results from numbering the intervals after sorting them on their right ends [2], can be obtained in linear time, that is, $O(m+n)$ time. An ordering of the vertices according to this numbering is found to be quite useful in solving many problems on interval graphs [2, 24]. Throughout the paper, the vertex numbered with i will be denoted by v_i, $1 \le i \le n$, and such an ordering will be denoted by π. We say that $v_i < v_j$ if $i < j$, $1 \le i, j \le n$.

Let G be an interval graph with vertex set $V(G)$ and edge set $E(G)$, $\mathcal{T}$ be a set containing a single vertex of $V(G)$, and let $\mathcal{P}_{\mathcal{T}}(G)$ be a minimum 1-fixed-endpoint path cover of G with respect to $\mathcal{T}$ of size $\lambda_{\mathcal{T}}(G)$ (or $\lambda_{\mathcal{T}}$ for short); recall that the size of $\mathcal{P}_{\mathcal{T}}(G)$ is the number of paths it contains. The vertex belonging to the set $\mathcal{T}$ is called *terminal* vertex, and the set $\mathcal{T}$ is called the *terminal set* of G, while those of $V(G) - \mathcal{T}$ are called *non-terminal* vertices. Thus, the set $\mathcal{P}_{\mathcal{T}}(G)$ contains two types of paths, which we call *terminal* and *non-terminal* paths: a *terminal path* P_t is a path having the terminal vertex as an endpoint and a *non-terminal path* P_f is a path having both its endpoints in $V(G) - \mathcal{T}$. The set of the non-terminal paths in a minimum 1PC of the graph G is denoted by N, while T denotes the set containing the terminal path. Clearly, $|T| = 1$ and $\lambda_{\mathcal{T}} = |N| + 1$.

Our algorithm for computing a 1PC of an interval graph is based on a greedy principle, visiting the vertices according to the ordering $\pi = (v_1, v_2, \ldots, v_k, \ldots, v_n)$, and uses three operations on the paths of a 1PC of $G[S]$, where $S = \{v_1, v_2, \ldots, v_k\}$, $1 \le k < n$. These three operations, namely `connect`, `insert` and `bridge` operations, are described below and are illustrated in Fig. 2:

- `Connect` operation. Let v_i be a non-terminal endpoint of a path P of $\mathcal{P}_{\mathcal{T}}(G[S])$ and let v_{k+1} be a non-terminal or a terminal vertex such that v_{k+1} sees v_i. We say that we *connect* vertex v_{k+1} to the path P, or, equivalently, to the vertex v_i, if we extend the path P by adding an edge which joins vertex v_{k+1} with vertex v_i.
- `Insert` operation. Let $P = (\ldots, v_i, v_j, \ldots)$, $i \ne j$, $i, j \in [1, k]$, be a path of $\mathcal{P}_{\mathcal{T}}(G[S])$ and let v_{k+1} be a non-terminal vertex such that v_{k+1} sees v_i and v_j. We say that we *insert* vertex v_{k+1} into P (between v_i and v_j), if we replace the path P with the path $P' = (\ldots, v_i, v_{k+1}, v_j, \ldots)$.

Fig. 2 Illustrating (**a**) `connect`, (**b**) `insert`, and (**c**) `bridge` operations; $P, P_1, P_2 \in \mathcal{P}_{\mathcal{T}}(G[S])$



- `Bridge` operation. Let P_1 and P_2 be two paths of $\mathcal{P}_{\mathcal{T}}(G[S])$ and let v_{k+1} be a non-terminal vertex that sees at least one non-terminal endpoint of P_1 and at least one of P_2. We say that we *bridge* the two paths P_1 and P_2 using vertex v_{k+1} if we connect v_{k+1} with a non-terminal endpoint of P_1 and a non-terminal endpoint of P_2.

Let P be a path of $\mathcal{P}_{\mathcal{T}}(G)$ and let v_i and v_j be its endpoints. We say that v_i is the left (resp. right) endpoint of the path and v_j is the right (resp. left) endpoint of the path if $v_i < v_j$ (resp. $v_j < v_i$). Throughout the paper, a trivial path (i.e. a path consisting of one vertex) is considered to have two endpoints, while a trivial path consisting of the terminal vertex $u \in \mathcal{T}$ is considered to have one terminal endpoint and one non-terminal endpoint.

Let G be an interval graph on n vertices and let $\mathcal{P}_{\mathcal{T}}(G)$ be a minimum 1PC of size $\lambda_{\mathcal{T}}$. Since a trivial path is considered to have two endpoints, the number of endpoints in $\mathcal{P}_{\mathcal{T}}(G)$ is $2\lambda_{\mathcal{T}}$. For each vertex v_i we denote by $d(v_i)$ the number of neighbors of v_i in $\mathcal{P}_{\mathcal{T}}(G)$; that is, $d(v_i) \in \{0, 1, 2\}$. We call *d-connectivity* of $\mathcal{P}_{\mathcal{T}}(G)$ the sum of $d(v_1), d(v_2), \ldots, d(v_n)$. It is easy to see that $\sum_{i=1}^{n} d(v_i) = 2(n - \lambda_{\mathcal{T}})$. Clearly, any minimum 1PC $\mathcal{P}_{\mathcal{T}}(G)$ has d-connectivity equal to $2(n - \lambda_{\mathcal{T}})$.

3 The Algorithm

In this section we present an algorithm for the 1PC problem on interval graphs. Our algorithm takes as input an interval graph G on n vertices and m edges and a set $\mathcal{T} = \{u\}$ containing the terminal vertex $u \in V(G)$, and computes a minimum 1PC $\mathcal{P}_{\mathcal{T}}(G)$ of G in $O(n^3)$ time. The algorithm is based on a greedy principle to extend a path of a minimum 1PC using operations on the left and right endpoints of its paths and properties of the graph $G[\{v_1, v_2, \ldots, v_i\} - \{u\}]$, $1 \le i \le n$; if a vertex sees the two endpoints of only one non-terminal path P, it is connected to the left endpoint of the path P. Furthermore, for each vertex v_i, $1 \le i < j$, we denote by $\varepsilon_i^{(j)}$ the number of endpoints v_κ belonging to different paths in $\mathcal{P}_{\mathcal{T}}(G[v_1, v_2, \ldots, v_j])$ with index $\kappa \in (i, j]$. We also define $\varepsilon_i^{(i)} = 0$ and $\varepsilon_0^{(i)} = \lambda_{\mathcal{T}}(G[v_1, v_2, \ldots, v_i])$, $1 \le i \le n$.

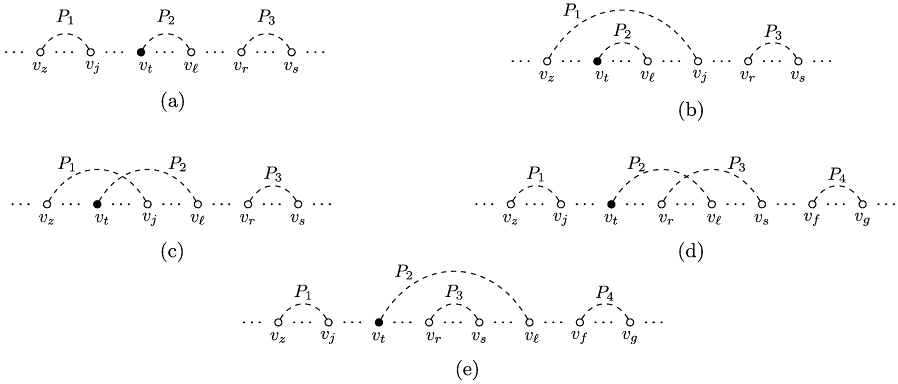


Fig. 3 Illustrating some cases of the `bridge` operation

Before describing our algorithm, let us present the operation `bridge` in detail. In most cases we bridge two paths that have the leftmost non-terminal endpoints. Suppose that when vertex v_i is processed it sees at least one non-terminal endpoint of a non-terminal path P_1, say, v_j, and at least the non-terminal endpoint of the terminal path P_2, say, v_ℓ, and both endpoints of a non-terminal path P_3, say, v_r and v_s. Let v_z, v_t and v_r be the left endpoints and v_j, v_ℓ and v_s be the right endpoints of the paths P_1, P_2 and P_3, respectively. There exist three cases where, in order to obtain the maximum possible value for every $\varepsilon_i^{(j)}$, we do not bridge two paths through the leftmost non-terminal endpoints (see Fig. 3(a)–(c)). In these three cases the bridge operation works as follows:

(a) $v_z < v_j < v_t < v_\ell < v_r < v_s$: we bridge P_1 and P_3 through v_z (or, v_j if $v_z \notin N(v_i)$) and v_r.
(b) $v_z < v_t < v_\ell < v_j < v_r < v_s$: if $v_z \notin N(v_i)$ we bridge P_2 and P_3 through v_ℓ and v_r; otherwise, we bridge P_1 and P_2 through v_z and v_ℓ.
(c) $v_z < v_t < v_j < v_\ell < v_r < v_s$: we bridge P_1 and P_3 through v_z (or, v_j if $v_z \notin N(v_i)$) and v_r.

Suppose now that P_1 is a non-terminal path having v_z and v_j as its left and right endpoints, respectively, P_2 is the terminal path with left endpoint v_t and right endpoint v_ℓ. Also, let P_3 be a non-terminal path with left and right endpoints v_r and v_s, respectively, and P_4 a non-terminal path with left and right endpoints v_f and v_g, respectively (see Fig. 3(d)–(e)). We distinguish the following two cases:

(d) $v_z < v_j < v_t < v_r < v_\ell < v_s < v_f < v_g$: if $v_z \in N(v_i)$ or $v_j \in N(v_i)$ we bridge P_1 and P_3 through v_z (v_j if $v_z \notin N(v_i)$) and v_r. If $v_\ell \in N(v_i)$ and $v_r \notin N(v_i)$ we bridge P_2 and P_4 through v_ℓ and v_f.
(e) $v_z < v_j < v_t < v_r < v_s < v_\ell < v_f < v_g$: if $v_z \in N(v_i)$ or $v_j \in N(v_i)$ we bridge P_1 and P_3 through v_i (v_j if $v_z \notin N(v_i)$) and v_r; otherwise, we bridge P_3 and P_4 through v_r (v_s if $v_r \notin N(v_i)$) and v_f.

Figure 3 presents cases (a)–(e). Suppose that we have the two paths P_2 and P_3 of case (e) and vertex v_i sees both v_r and v_s, that is, $P_2 = (v_t, \ldots, v_a, v_b, v_c, \ldots, v_\ell)$

and $P_3 = (v_r, \ldots, v_s)$, where $v_a < v_s < v_b$ and $v_s < v_c$. Then, the bridge operation constructs the path $P = (v_t, \ldots, v_a, v_b, v_s, \ldots, v_r, v_i, v_c, \ldots, v_\ell)$. Suppose now that we have the two paths P_1 and P_2 of case (c) and vertex v_i sees all vertices with index greater or equal to z, that is, $P_1 = (v_z, \ldots, v_j)$ and $P_2 = (v_t, \ldots, v_a, v_b, v_c, \ldots, v_\ell)$, where $v_a < v_j < v_b$ and $v_j < v_c$. Then, the bridge operation constructs the path $P = (v_t, \ldots, v_a, v_b, v_j, \ldots, v_z, v_i, v_c, \ldots, v_\ell)$. Suppose that there exist two paths P_2 and P_3 as in case (d) and vertex v_i sees all vertices with index k, $d \le k$, where $r < d \le \ell$, that is, $P_2 = (v_t, \ldots, v_a, v_b, v_c, \ldots, v_\ell)$ and $P_3 = (v_r, \ldots, v_s)$, where $v_a < v_r < v_b$ and $v_r < v_c$. If $d < c$ then the bridge operation constructs the path $P = (v_t, \ldots, v_a, v_b, v_r, \ldots, v_s, v_i, v_c, \ldots, v_\ell)$; otherwise, it constructs the path $P = (v_t, \ldots, v_a, v_b, v_r, \ldots, v_s, v_i, v_\ell, \ldots, v_c)$. If there exist two paths P_1 and P_2 as in case (b) and vertex v_i sees all vertices with index greater or equal to z, that is, $P_1 = (v_z, \ldots, v_a, v_b, v_c, \ldots, v_j)$ and $P_2 = (v_t, \ldots, v_\ell)$, where $v_a < v_\ell < v_b$ and $v_\ell < v_c$, then the bridge operation constructs the path $P = (v_t, \ldots, v_\ell, v_b, v_a, \ldots, v_z, v_i, v_c, \ldots, v_j)$, if $c < j$, or the path $P = (v_t, \ldots, v_\ell, v_b, v_a, \ldots, v_z, v_i, v_j, \ldots, v_c)$, if $j < c$.

We next describe the operation `new_path` of our algorithm which creates a new path when the vertex v_i is processed. There exist three cases where operation new_path creates a new non-trivial path while in all other cases it creates a new trivial path. Suppose that v_i sees an internal vertex v_j belonging to a path $P = (v_s, \ldots, v_r, v_j, v_\ell, \ldots, v_t)$ such that $v_r < v_s < v_t < v_\ell < v_j$. We remove the edge $v_j v_\ell$ from P and we obtain $P_1 = (v_s, \ldots, v_r, v_j)$ and $P_2 = (v_t, \ldots, v_\ell)$. Then, we connect v_i to v_j. The case where $v_j v_s \in E(G)$ and $v_j v_r \notin E(G)$ is similar. If v_i sees an internal vertex v_j belonging to a path $P = (v_r, \ldots, v_s, v_j, v_\ell, \ldots, v_t)$ such that $v_r < v_t < v_\ell < v_s < v_j$, we remove the edge $v_s v_j$ from P and we obtain $P_1 = (v_r, \ldots, v_s)$ and $P_2 = (v_t, \ldots, v_\ell, v_j)$. Then, we connect v_i to v_j. Suppose now that v_i sees an internal vertex v_j belonging to a path $P = (v_\ell, \ldots, v_s, v_j, v_r, \ldots, v_t)$ such that $v_\ell < v_t < v_s < v_r < v_j$. We remove the edge $v_j v_r$ from P and we obtain $P_1 = (v_\ell, \ldots, v_s, v_j)$ and $P_2 = (v_t, \ldots, v_r)$. Then, we connect v_i to v_j. The above cases, where the operation new_path creates a new non-trivial path, are described below:

(a) $v_r < v_s < v_t < v_\ell < v_j$. We create paths $P_1 = (v_s, \ldots, v_r, v_j, v_i)$ and $P_2 = (v_t, \ldots, v_\ell)$. The case where $v_j v_s \in E(G)$ and $v_j v_r \notin E(G)$ is similar.
(b) $v_r < v_t < v_\ell < v_s < v_j$. We create paths $P_1 = (v_r, \ldots, v_s)$ and $P_2 = (v_t, \ldots, v_\ell, v_j, v_i)$.
(c) $v_\ell < v_t < v_s < v_r < v_j$. We create paths $P_1 = (v_\ell, \ldots, v_s, v_j, v_i)$ and $P_2 = (v_t, \ldots, v_r)$.

Note that, the rightmost endpoint of a path in $\mathcal{P}_\mathcal{T}(G[\{v_1, v_2, \ldots, v_{i-1}\}])$ is v_t and, thus, $\varepsilon_t^{(i-1)} = 0$. The 1PC $\mathcal{P}_\mathcal{T}(G)$ of the graph G in each of the above cases contains two new endpoints, vertices v_i and $v_{k'}$ such that $t < k'$; thus, $\varepsilon_t^{(i)} = 2$. Figure 4 presents the above cases.

The operation `connect_break` of our algorithm is similar to the operation new_path. Specifically, suppose that in the above cases (a)–(c) there exists a path $P = (v_a, \ldots, v_b)$ such that $v_j < v_a < v_b < v_i$. Then, the operation connect_break works similarly to the operation new_path; the only difference is that v_i is also connected to v_a.

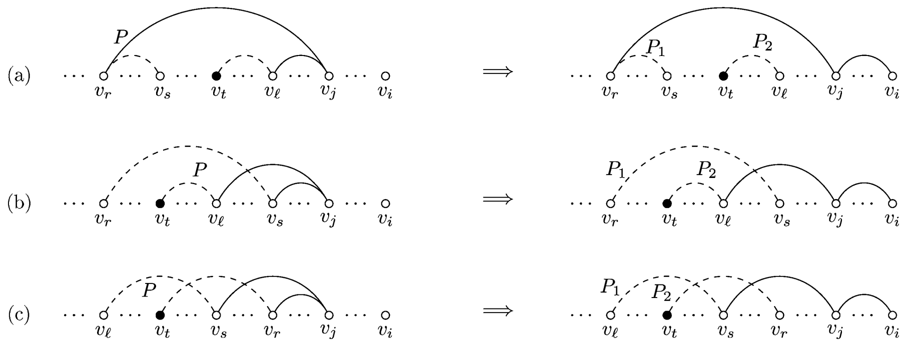


Fig. 4 Illustrating some cases of the `new_path` operation

We next present our algorithm which works as follows.

Algorithm 1PC_INTERVAL
Input: an interval graph G on n vertices and m edges and a vertex $u \in V(G)$;
Output: a minimum 1PC $\mathcal{P}_{\mathcal{T}}(G)$ of the interval graph G, where $\mathcal{T} = \{u\}$;

1. Construct the ordering π of the vertices of G. Let t be the index of the terminal vertex u;
2. $\lambda_{\mathcal{T}} \leftarrow 1$; $P_{\lambda_{\mathcal{T}}} \leftarrow (v_1)$; $\varepsilon_0^{(1)} \leftarrow 1$;
   v_1 is the left and right endpoint of the path $P_{\lambda_{\mathcal{T}}}$;
3. `for` $i = 2$ `to` n `do`
   `if` $i \neq t$ `then` execute the procedure `process_non_terminal`;
   `else` execute the procedure `process_terminal`;
4. $\mathcal{P}_{\mathcal{T}}(G) = \{P_1, P_2, \dots, P_{\lambda_{\mathcal{T}}}\}$;

End_of_Algorithm 1PC_INTERVAL.

The procedures `process_non_terminal` and `process_terminal` are described in Figs. 5 and 6, respectively. We point out that, if no vertex is specified as the terminal vertex then Algorithm 1PC_INTERVAL returns a minimum path cover of G.

We next show that, if $\lambda_{\mathcal{T}}(G)$ is the size of a minimum 1PC of G with respect to $\mathcal{T} = \{v_t\}$ then the size of a minimum 1PC of $G - \{v_t\}$ is either $\lambda_{\mathcal{T}}(G)$ or $\lambda_{\mathcal{T}}(G) - 1$.

Lemma 3.1 *Let G be an interval graph and $\lambda_{\mathcal{T}}(G)$ be the size of a minimum* 1PC *of G with respect to $\mathcal{T} = \{v_t\}$. The size of a minimum PC of $G - \{v_t\}$ is either $\lambda_{\mathcal{T}}(G)$ or $\lambda_{\mathcal{T}}(G) - 1$.*

Proof Suppose that the size of a minimum PC of $G - \{v_t\}$ is at least $\lambda_{\mathcal{T}}(G) + 1$. Since a terminal vertex cannot decrease the size of a minimum 1PC, we have $\lambda_{\mathcal{T}}(G) \geq \lambda_{\mathcal{T}}(G - \{v_t\})$. Thus, $\lambda_{\mathcal{T}}(G) \geq \lambda_{\mathcal{T}}(G) + 1$, a contradiction. Suppose now that the size of a minimum PC $\mathcal{P}_{\mathcal{T}}(G - \{v_t\})$ of $G - \{v_t\}$ is at most $\lambda_{\mathcal{T}}(G) - 2$. Then, adding a trivial path containing vertex v_t to $\mathcal{P}_{\mathcal{T}}(G - \{v_t\})$ results to a 1PC of G of size $\lambda_{\mathcal{T}}(G) - 1$, a contradiction. $\square$

`Process_non_terminal`

{the vertex v_i of $\pi = (v_1, v_2, \ldots, v_i, \ldots, v_n)$ is not terminal, that is, $i \neq t$}
{let $N'(v_i) = \{v_j \in N(v_i) : j < i\}$, v_j is the leftmost neighbor of $v_i \in N'(v_i)$}

- `if` $N'(v_i) \neq \emptyset$ `and` $\varepsilon_{j-1}^{(i-1)} \geq 2$ `then`
    `if` at least two endpoints are non-terminal vertices `then` `bridge`; $\lambda_{\mathcal{T}} \leftarrow \lambda_{\mathcal{T}} - 1$;
    `else` {$\varepsilon_{j-1}^{(i-1)} = 2$ *and one endpoint is the terminal vertex* v_t.}
      `if` 1PC_INTERVAL on $G[\{v_1, \ldots, v_{i-1}\} - \{v_t\}]$ returns a path cover of $\lambda_{\mathcal{T}} - 1$ paths
      `then` {*let* $\mathcal{P}_{\mathcal{T}}(G[\{v_1, \ldots, v_{i-1}\} - \{v_t\}])$ *be such a path cover*}
        `connect` v_i to the leftmost endpoint of $\mathcal{P}_{\mathcal{T}}(G[\{v_1, \ldots, v_{i-1}\} - \{v_t\}])$;
        `connect` v_t to v_i;
        $\lambda_{\mathcal{T}} \leftarrow \lambda_{\mathcal{T}} - 1$;
      `else-if` 1PC_INTERVAL on $G[\{v_1, \ldots, v_{i-1}\} - \{v_t\}]$ returns a path cover of $\lambda_{\mathcal{T}}$
        paths `then`
      `if` during the execution of 1PC_INTERVAL on $G[\{v_1, \ldots, v_{i-1}\}]$ and v_t
        the following hold:

        (i) there exists v_b such that $v_t \in N'(v_b)$;
        (ii) breaking v_t from its path and connecting v_t to v_b results to a 1PC $\mathcal{P}'_{\mathcal{T}}(G[\{v_1, \ldots, v_{i-1}\}])$ of $\lambda_{\mathcal{T}}$ paths;
        (iii) $\varepsilon_{j-1}^{(i-1)} = 2$ and both endpoints are non-terminal vertices

      `then` $\mathcal{P}_{\mathcal{T}}(G[\{v_1, \ldots, v_{i-1}\}]) \leftarrow \mathcal{P}'_{\mathcal{T}}(G[\{v_1, \ldots, v_{i-1}\}])$; `bridge`; $\lambda_{\mathcal{T}} \leftarrow \lambda_{\mathcal{T}} - 1$;
      `else` `connect_break`;
- `if` $N'(v_i) \neq \emptyset$ `and` $\varepsilon_{j-1}^{(i-1)} = 1$ `and` the endpoint v_f, $j \leq f \leq i-1$, is non terminal `then`
    `if` v_i sees an internal vertex v_s `then` `connect_break`;
    `else` `connect` v_i to the leftmost non-terminal endpoint;
- `if` $N'(v_i) = \emptyset$ `or` $\varepsilon_{j-1}^{(i-1)} = 0$ `or`
    ($\varepsilon_{j-1}^{(i-1)} = 1$ `and` the endpoint v_t, $j \leq t \leq i-1$, is the terminal vertex) `then`
    `if` v_i has two consecutive neighbors into a path `then` `insert` v_i into the path;
    `else-if` v_i sees an internal vertex v_s `then`
      `if` $v_s v_a$ is an edge of a path P_k and v_a sees an endpoint v_b of a path $P_{k'}$, $k \neq k'$
      `then` remove the edge $v_s v_a$ of P; `connect` v_a to v_b; `connect` v_i to v_j;
      `else` `new_path`; $\lambda_{\mathcal{T}} \leftarrow \lambda_{\mathcal{T}} + 1$;
    `else` $\lambda_{\mathcal{T}} \leftarrow \lambda_{\mathcal{T}} + 1$; $P_{\lambda_{\mathcal{T}}} \leftarrow (v_i)$;
- `update` the values $\varepsilon_k^{(i)}$, $0 \leq k \leq i$, and the left and right endpoints of the paths;

Fig. 5 The procedure `process_non_terminal` of the Algorithm 1PC_INTERVAL

`Process_terminal`

{the vertex v_i of $\pi = (v_1, v_2, \ldots, v_i, \ldots, v_n)$ is terminal, that is, $i = t$}

- `if` $\varepsilon_{j-1}^{(i-1)} \geq 1$ `then` `connect` v_i to the leftmost endpoint of $\mathcal{P}_{\mathcal{T}}(G[\{v_1, \ldots, v_{i-1}\}])$;
    `else` $\lambda_{\mathcal{T}} \leftarrow \lambda_{\mathcal{T}} + 1$; $P_{\lambda_{\mathcal{T}}} \leftarrow (v_i)$;
- `update` the values $\varepsilon_k^{(i)}$, $0 \leq k \leq i$, and the left and right endpoints of the paths;

Fig. 6 The procedure `process_terminal` of the Algorithm 1PC_INTERVAL

Concerning the ordering of the endpoints of the paths of the 1PC constructed by Algorithm 1PC_INTERVAL, we prove the following lemma.

Lemma 3.2 *Let G be an interval graph with no terminal vertex. Let P_s, $1 \leq s \leq \lambda_{\mathcal{T}}$, be a path in the* 1PC *$\mathcal{P}_{\mathcal{T}}(G)$ of the graph G constructed by Algorithm* 1PC_INTERVAL *and let v_i and v_j be the left and right endpoints of P_s, respectively. Then, there is no path $P_t \in \mathcal{P}_{\mathcal{T}}(G)$, $1 \leq t \leq \lambda_{\mathcal{T}}$, $t \neq s$, such that $v_i < v_k < v_j$ or $v_i < v_\ell < v_j$, where v_k and v_ℓ are the left and right endpoints of P_t, respectively.*

Proof Let P_s, $1 \leq s \leq \lambda_{\mathcal{T}}$ be a path in the 1PC $\mathcal{P}_{\mathcal{T}}(G)$ constructed by Algorithm 1PC_INTERVAL and let v_i and v_j be its left and right endpoints, respectively. Let $P_t \in \mathcal{P}_{\mathcal{T}}(G)$, $1 \leq t \leq \lambda_{\mathcal{T}}$, $t \neq s$, and let v_k and v_ℓ be its left and right endpoints, respectively. Suppose that $v_i < v_k < v_j$. Since v_i and v_j are the endpoints of P_s and $v_i < v_k < v_j$, the path P_s contains at least one edge, say, $v_a v_b$, such that $v_a < v_k < v_b$. Clearly, vertices v_a and v_b are non-terminal vertices. Since $v_a v_b \in E(G)$, we also have $v_k v_b \in E(G)$. Then, according to Algorithm 1PC_INTERVAL, when vertex v_b is processed, vertex v_k is an endpoint of the path P_t, and, thus, v_b bridges the paths P_s and P_t through vertices v_a and v_k, a contradiction. Similarly, we can prove that $v_\ell < v_i$ or $v_\ell > v_j$. □

Using similar arguments we can prove the following lemma.

Lemma 3.3 *Let G be an interval graph containing a terminal vertex. Let P_s, $1 \leq s \leq \lambda_{\mathcal{T}}$, be a non-terminal path in the* 1PC *$\mathcal{P}_{\mathcal{T}}(G)$ of the graph G constructed by Algorithm* 1PC_INTERVAL *and let v_i and v_j be the left and right endpoints of P_s, respectively. Then, there is no non-terminal path $P_t \in \mathcal{P}_{\mathcal{T}}(G)$, $1 \leq t \leq \lambda_{\mathcal{T}}$, $t \neq s$, such that $v_i < v_k < v_j$ or $v_i < v_\ell < v_j$, where v_k and v_ℓ are the left and right endpoints of P_t, respectively.*

4 Correctness and Time Complexity

Let G be an interval graph on n vertices and m edges and let $\mathcal{T}$ be a subset of $V(G)$ containing a single vertex $v \in V(G)$. In order to prove the correctness of Algorithm 1PC_INTERVAL, we use induction on n. We also prove a property of the minimum 1PC $\mathcal{P}_{\mathcal{T}}(G)$ of G constructed by our algorithm, namely, *Property* 1PC-*on-H*:

Property 1PC-on-H *Let H be an interval graph with vertex set $V(H) = \{u_1, u_2, \ldots, u_n\}$, let $\mathcal{P}_{\mathcal{T}}(H)$ be a minimum* 1PC *of the graph H, and let $\varepsilon_i^{(n)}$ be the number of endpoints v_κ belonging to different paths with index $\kappa \in (i, n]$, $1 \leq i \leq n$.*

The minimum 1PC *$\mathcal{P}_{\mathcal{T}}(H)$ satisfies Property* 1PC-*on-H if there is no other minimum* 1PC *$\mathcal{P}'_{\mathcal{T}}(H)$ having $\varepsilon_i'^{(n)}$ endpoints $v_{\kappa'}$ belonging to different paths with index $\kappa' \in (i, n]$ such that $\varepsilon_i'^{(n)} > \varepsilon_i^{(n)}$, $1 \leq i < \varrho$, where ϱ is the index of the rightmost endpoint of a path in $\mathcal{P}_{\mathcal{T}}(H)$, and exactly one of the following holds*:

(i) $\varepsilon_i'^{(n)} \le \varepsilon_i^{(n)}$, $\varrho \le i \le n$;
(ii) $\varepsilon_i'^{(n)} = \varepsilon_i^{(n)} + 1$, $\varrho \le i < \varrho'$ *and* $\varepsilon_i'^{(n)} = \varepsilon_i^{(n)}$, $\varrho' \le i \le n$, *where* ϱ' *is the index of the rightmost endpoint of a path in* $\mathcal{P}'_{\mathcal{T}}(H)$, *and there exists a vertex* v_z *such that* $\varepsilon_z^{(n)} > \varepsilon_z'^{(n)}$ *and there exists no vertex* $v_{z'}$ *such that* $\varepsilon_{z'}'^{(n)} > \varepsilon_{z'}^{(n)}$, $1 \le z, z' < \varrho$.

Recall that a trivial path has two endpoints that coincide. We next prove Lemmas 4.1–4.6 that concern the operations performed by the Algorithm 1PC_INTERVAL when vertex v_n is processed. These lemmas make the proof of Theorem 4.1 easier to follow.

Lemma 4.1 *Let G be an interval graph and suppose that Algorithm* 1PC_Interval *computes a minimum* 1PC $\mathcal{P}_{\mathcal{T}}(G[S])$ *of the graph* $G[S]$, $S = \{v_1, v_2, \ldots, v_{n-1}\}$, *on at most* $n-1$ *vertices satisfying Property* 1PC-on-$G[S]$. *Let* $\lambda_{\mathcal{T}}(G[S])$ *be the size of* $\mathcal{P}_{\mathcal{T}}(G[S])$ *and let* v_n *be a non-terminal vertex. If the algorithm uses* v_n *to bridge two paths* (*operation bridge*), *then Algorithm* 1PC_Interval *computes a minimum* 1PC $\mathcal{P}_{\mathcal{T}}(G)$ *of the graph* G *satisfying Property* 1PC-on-G.

Proof Let $\lambda_{\mathcal{T}}(G)$ be the size of $\mathcal{P}_{\mathcal{T}}(G)$. Clearly, the size $\lambda'_{\mathcal{T}}(G)$ of a minimum 1PC of G is equal to $\lambda_{\mathcal{T}}(G[S]) - 1$ or $\lambda_{\mathcal{T}}(G[S])$ or $\lambda_{\mathcal{T}}(G[S]) + 1$. When the algorithm processes vertex v_n, it uses v_n to bridge two paths (operation bridge), that is, $\lambda_{\mathcal{T}}(G) = \lambda_{\mathcal{T}}(G[S]) - 1$; consequently, $\mathcal{P}_{\mathcal{T}}(G)$ is a minimum 1PC of G, that is, $\lambda'_{\mathcal{T}}(G) = \lambda_{\mathcal{T}}(G)$.

Algorithm 1PC_INTERVAL computes a minimum 1PC $\mathcal{P}_{\mathcal{T}}(G[S])$ of the interval graph $G[S]$, $S = \{v_1, v_2, \ldots, v_{n-1}\}$, on at most $n-1$ vertices having $\varepsilon_i^{(n-1)}$ endpoints v_κ belonging to different paths with index $\kappa \in (i, n-1]$, $1 \le i \le n-1$, such that there is no other minimum 1PC $\mathcal{P}'_{\mathcal{T}}(G[S])$ having $\varepsilon_i'^{(n-1)}$ endpoints $v_{\kappa'}$ belonging to different paths with index $\kappa' \in (i, n-1]$ such that $\varepsilon_i'^{(n-1)} > \varepsilon_i^{(n-1)}$, $1 \le i < d$, where d is the index of the rightmost endpoint of a path in $\mathcal{P}_{\mathcal{T}}(G[S])$, and exactly one of the following holds:

(i) $\varepsilon_i'^{(n-1)} \le \varepsilon_i^{(n-1)}$, $d \le i \le n-1$;
(ii) $\varepsilon_i'^{(n-1)} = \varepsilon_i^{(n-1)} + 1$, $d \le i < d'$ and $\varepsilon_i'^{(n-1)} = \varepsilon_i^{(n-1)}$, $d' \le i \le n-1$, where d' is the index of the rightmost endpoint of a path in $\mathcal{P}'_{\mathcal{T}}(G[S])$, and there exists a vertex v_q such that $\varepsilon_q^{(n-1)} > \varepsilon_q'^{(n-1)}$, and there exists no vertex $v_{q'}$ such that $\varepsilon_{q'}'^{(n-1)} > \varepsilon_{q'}^{(n-1)}$, $1 \le q, q' < d$.

We distinguish the following cases:

Case 1. Suppose that $\varepsilon_i'^{(n-1)} \le \varepsilon_i^{(n-1)}$, $d \le i \le n-1$. We show that the algorithm computes a minimum 1PC $\mathcal{P}_{\mathcal{T}}(G)$ of the graph G having $\varepsilon_i^{(n)}$ endpoints v_κ belonging to different paths with index $\kappa \in (i, n]$, $1 \le i \le n$, such that there is no other minimum 1PC $\mathcal{P}'_{\mathcal{T}}(G)$ having $\varepsilon_i'^{(n)}$ endpoints $v_{\kappa'}$ belonging to different paths with index $\kappa' \in (i, n]$ such that $\varepsilon_i'^{(n)} > \varepsilon_i^{(n)}$, $1 \le i \le n$. Clearly, vertex v_n is an internal vertex of a path in any other minimum 1PC $\mathcal{P}'_{\mathcal{T}}(G)$, otherwise removing it from $\mathcal{P}'_{\mathcal{T}}(G)$ would result to a 1PC of $G[S]$ of size $\le \lambda_{\mathcal{T}}(G[S]) - 1$, a contradiction. Assume that there exists a minimum 1PC $\mathcal{P}'_{\mathcal{T}}(G)$ having an index, say, $k-1$, for which we have

$\varepsilon'^{(n)}_{k-1}$ endpoints $v_{\kappa'}$ belonging to different paths with index $\kappa' \in (k-1, n]$ such that $\varepsilon'^{(n)}_{k-1} > \varepsilon^{(n)}_{k-1}$. Suppose that $\varepsilon'^{(n)}_{k-1} - \varepsilon^{(n)}_{k-1} = 1$. Note that $\varepsilon^{(n)}_1 \geq \varepsilon'^{(n)}_1$. Indeed, let $\mathcal{P}'_{\mathcal{T}}(G)$ be a minimum 1PC of G having $\varepsilon'^{(n)}_1 = \varepsilon^{(n)}_1 + 1$ endpoints $v_{\kappa'}$ belonging to different paths with index $\kappa' > 1$. Suppose that vertex v_1 is an internal vertex in $\mathcal{P}_{\mathcal{T}}(G)$. Then, the algorithm constructs $\varepsilon^{(n)}_1$ paths while $\mathcal{P}'_{\mathcal{T}}(G)$ contains at least $\varepsilon^{(n)}_1 + 1$ paths, a contradiction. Suppose that vertex v_1 is an endpoint in $\mathcal{P}_{\mathcal{T}}(G)$. Note that, according to the algorithm, if v_1 has degree greater or equal to one, then it belongs to a path containing more than one vertex. Consequently, the other endpoint of the path containing v_1 is one of the $\varepsilon^{(n)}_1$ endpoints, and, thus, the algorithm constructs $\varepsilon^{(n)}_1$ paths while $\mathcal{P}'_{\mathcal{T}}(G)$ contains at least $\varepsilon^{(n)}_1 + 1$ paths, a contradiction. Since $\varepsilon^{(n)}_1 \geq \varepsilon'^{(n)}_1$ there exists a vertex v_j, $1 \leq j < k-1$, such that $\varepsilon^{(n)}_j = \varepsilon'^{(n)}_j$. This implies that vertex v_{j+1} is the right endpoint of a path P in the minimum 1PC $\mathcal{P}_{\mathcal{T}}(G)$ constructed by the algorithm. Without loss of generality, we assume that $\varepsilon'^{(n)}_i = \varepsilon^{(n)}_i + 1$, $j+1 \leq i < k-1$.

Let $P' = (\ldots, v_a, v_n, v_b, \ldots)$ be the path of $\mathcal{P}'_{\mathcal{T}}(G)$ containing vertex v_n. Then, $j+1 < a$ and $j+1 < b$. Indeed, suppose that at least one of v_a and v_b has index less or equal to $j+1$, say, $a < j+1$. Since $v_a v_n \in E(G)$ we have $v_{j+1} v_n \in E(G)$. Let $P = (\ldots, v_c, v_n, v_d, \ldots)$ be the path of $\mathcal{P}_{\mathcal{T}}(G)$ containing vertex v_n. Suppose that $v_c < v_{j+1}$ and $v_d < v_{j+1}$. Then, due to the induction hypothesis, both of the endpoints of P have index greater than $j+1$. However, according to the algorithm (operation bridge), such an ordering of the endpoints cannot exist, a contradiction. Suppose that $v_c > v_{j+1}$ and $v_d > v_{j+1}$. Then, due to the induction hypothesis, at least one of the endpoints of P should have index greater than $j+1$. Again, according to the algorithm (operation bridge), such an ordering of the endpoints cannot exist, a contradiction. Suppose now that $v_c < v_{j+1}$ and $v_d > v_{j+1}$. Then, due to the induction hypothesis, computing a 1PC of $G[S]$ we had case (e) which is described in Sect. 3 and $v_{j+1} = v_s$. However, in this case, vertex v_{j+1} would not be the right endpoint of a path in $\mathcal{P}_{\mathcal{T}}(G[S])$, a contradiction.

Consequently, $j+1 < a$ and $j+1 < b$. Then, both endpoints of P have indexes less than $j+1$. This implies that vertex v_n has bridged two non-terminal paths for which we have an endpoint of one path between the endpoints of the other, which is a contradiction according to Lemma 3.3.

Consequently, we have shown that there does not exist a minimum 1PC $\mathcal{P}'_{\mathcal{T}}(G)$ having an index, say, $k-1$, for which we have $\varepsilon'^{(n)}_{k-1}$ endpoints $v_{\kappa'}$ belonging to different paths with index $\kappa' \in (k-1, n]$, where $\varepsilon'^{(n)}_{k-1} > \varepsilon^{(n)}_{k-1}$.

Case 2. Suppose that $\varepsilon'^{(n-1)}_i = \varepsilon^{(n-1)}_i + 1$, $d \leq i < d'$ and $\varepsilon'^{(n-1)}_i = \varepsilon^{(n-1)}_i$, $d' \leq i \leq n-1$, where d' is the index of the rightmost endpoint of a path in $\mathcal{P}'_{\mathcal{T}}(G[S])$, and there exists a vertex v_q such that $\varepsilon^{(n-1)}_q > \varepsilon'^{(n-1)}_q$, and there exists no vertex v'_q such that $\varepsilon'^{(n-1)}_q > \varepsilon^{(n-1)}_q$, $1 \leq q, q' < d$. We show that the algorithm computes a minimum 1PC $\mathcal{P}_{\mathcal{T}}(G)$ of the graph G having $\varepsilon^{(n)}_i$ endpoints v_κ belonging to different paths with index $\kappa \in (i, n]$, $1 \leq i \leq n$, such that there is no other minimum 1PC $\mathcal{P}'_{\mathcal{T}}(G)$ having $\varepsilon'^{(n)}_i$ endpoints $v_{\kappa'}$ belonging to different paths with index $\kappa' \in (i, n]$, $1 \leq i \leq \varrho$, such that $\varepsilon'^{(n)}_i > \varepsilon^{(n)}_i$, $1 \leq i < \varrho$, where ϱ is the index of the rightmost endpoint of a path in $\mathcal{P}_{\mathcal{T}}(G)$, and exactly one of the following holds:

(i) $\varepsilon_i^{\prime(n)} \leq \varepsilon_i^{(n)}$, $\varrho \leq i \leq n$;

(ii) $\varepsilon_i^{\prime(n)} = \varepsilon_i^{(n)} + 1$, $\varrho \leq i < \varrho'$ and $\varepsilon_i^{\prime(n)} = \varepsilon_i^{(n)}$, $\varrho' \leq i \leq n$, where ϱ' is the index of the rightmost endpoint of a path in $\mathcal{P}'_{\mathcal{T}}(G)$, and there exists a vertex v_z such that $\varepsilon_z^{(n)} > \varepsilon_z^{\prime(n)}$, and there exists no vertex v'_z such that $\varepsilon_z^{\prime(n)} > \varepsilon_z^{(n)}$, $1 \leq z, z' < \varrho$.

Suppose that there exists a minimum 1PC $\mathcal{P}'_{\mathcal{T}}(G)$ such that $\varepsilon_\varrho^{(n)} = \varepsilon_\varrho^{\prime(n)} = 0$. Then, similarly to Case 1, we show that there is no other minimum 1PC $\mathcal{P}'_{\mathcal{T}}(G)$ such that $\varepsilon_i^{\prime(n)} > \varepsilon_i^{(n)}$, $1 \leq i < n$.

Suppose now that there exists a minimum 1PC $\mathcal{P}'_{\mathcal{T}}(G)$ such that $\varepsilon_i^{\prime(n)} = \varepsilon_i^{(n)} + 1$, $\varrho \leq i < \varrho'$ and $\varepsilon_i^{\prime(n)} = \varepsilon_i^{(n)}$, $\varrho' \leq i \leq n$, where ϱ' is the index of the rightmost endpoint of a path in $\mathcal{P}'_{\mathcal{T}}(G)$. We show that there exists a vertex v_z such that $\varepsilon_z^{(n)} > \varepsilon_z^{\prime(n)}$, $1 \leq z < \varrho$ and there exists no vertex $v_{z'}$, $1 \leq z' < \varrho$, such that $\varepsilon_{z'}^{\prime(n)} > \varepsilon_{z'}^{(n)}$.

Note that, there cannot exist a minimum 1PC $\mathcal{P}'_{\mathcal{T}}(G)$ such that $\varepsilon_i^{\prime(n)} = \varepsilon_i^{(n)} + 2$, $\varrho \leq i < \varrho'$. Indeed, suppose that there exists a minimum 1PC $\mathcal{P}'_{\mathcal{T}}(G)$ such that $\varepsilon_i^{\prime(n)} = \varepsilon_i^{(n)} + 2$, $\varrho \leq i < \varrho'$. Consider the case where v_n belongs to a path in $\mathcal{P}'_{\mathcal{T}}(G)$ and it is connected to vertices $v_{a'}$ and $v_{b'}$ such that $\varrho < a'$ and $\varrho < b'$. Then, removing v_n from $\mathcal{P}'_{\mathcal{T}}(G)$ we obtain a minimum 1PC of $G[S]$ such that $\varepsilon_\varrho^{\prime(n-1)} = \varepsilon_\varrho^{\prime(n)} + 1 = \varepsilon_\varrho^{(n)} + 3$ or $\varepsilon_\varrho^{\prime(n-1)} = \varepsilon_\varrho^{\prime(n)} + 2 = \varepsilon_\varrho^{(n)} + 4$. If v_n belongs to a path in $\mathcal{P}_{\mathcal{T}}(G)$ and it is connected to vertices v_a and v_b, then removing v_n from $\mathcal{P}_{\mathcal{T}}(G)$ we obtain a minimum 1PC of $G[S]$ such that $\varepsilon_\varrho^{(n-1)} \leq \varepsilon_\varrho^{(n)} + 2$. Thus, there exist three paths in $\mathcal{P}_{\mathcal{T}}(G)$ such that there are no left endpoints between their right endpoints in π, a contradiction. Consider now the case where v_n belongs to a path in $\mathcal{P}'_{\mathcal{T}}(G)$ and it is connected to vertices $v_{a'}$ and $v_{b'}$ such that $\varrho < a'$ and $\varrho > b'$. Then, $v_n v_\varrho \in E(G)$. Note that v_n belongs to a path P in $\mathcal{P}_{\mathcal{T}}(G)$ and it is connected to vertices v_a and v_b such that $a < \varrho$ and $\varrho < b$. Removing v_n from $\mathcal{P}_{\mathcal{T}}(G)$ should result to a minimum 1PC of $G[S]$ such that $\varepsilon_\varrho^{(n-1)} = \varepsilon_\varrho^{(n)} + 1$. Consequently, both endpoints of P have index less than ϱ, which implies that there exists a specific ordering of the endpoints of the paths in $\mathcal{P}_{\mathcal{T}}(G[S])$, which, according to the algorithm, is not possible, a contradiction. The case where v_n belongs to a path in $\mathcal{P}'_{\mathcal{T}}(G)$ and it is connected to vertices $v_{a'}$ and $v_{b'}$ such that $\varrho > a'$ and $\varrho > b'$ is similar.

Using similar arguments with Case 1, we show that there exists no vertex $v_{z'}$, $1 \leq z' < \varrho$, such that $\varepsilon_{z'}^{\prime(n)} > \varepsilon_{z'}^{(n)}$. Suppose that $\varepsilon_i^{(n)} = \varepsilon_i^{\prime(n)}$, $1 \leq i < \varrho$. Note that v_n belongs to a path $P = (v_\ell, \ldots, v_a, v_n, v_b, \ldots, v_r) \in \mathcal{P}_{\mathcal{T}}(G)$ such that $a > \varrho$ and $b > \varrho$. Thus, v_n belongs to a path $P' = (v_{\ell'}, \ldots, v_{a'}, v_n, v_{b'}, \ldots, v_{r'}) \in \mathcal{P}'_{\mathcal{T}}(G)$ such that $a' > \varrho$ and $b' > \varrho$. If we remove v_n from $\mathcal{P}'_{\mathcal{T}}(G)$ then there exists a vertex $v_z = v_{b'-1}$ such that $\varepsilon_z^{\prime(n-1)} = \varepsilon_z^{\prime(n)} + 1$. This implies that $r' > b'$. Furthermore, if we remove v_n from $\mathcal{P}_{\mathcal{T}}(G)$ and we obtain $\varepsilon_z^{(n-1)} = \varepsilon_z^{(n)} + 1$, which implies that $b = b'$, then $\varepsilon_i^{(n-1)} = \varepsilon_i^{\prime(n-1)}$, $1 \leq i < \varrho$, a contradiction. If we remove v_n from $\mathcal{P}'_{\mathcal{T}}(G)$ and we obtain $\varepsilon_z^{(n-1)} = \varepsilon_z^{(n)} + 2$, then there exists a specific ordering of the endpoints of the paths in $\mathcal{P}_{\mathcal{T}}(G[S])$ which, according to the algorithm, is not possible, a contradiction. Consequently, there exists a vertex v_z such that $\varepsilon_z^{(n)} > \varepsilon_z^{\prime(n)}$, $1 \leq z < \varrho$. $\square$

Lemma 4.2 *Let G be an interval graph and suppose that Algorithm* 1PC_Interval *computes a minimum* 1PC $\mathcal{P}_{\mathcal{T}}(G[S])$ *of the graph* $G[S]$, $S = \{v_1, v_2, \ldots, v_{n-1}\}$, *on at most* $n-1$ *vertices satisfying Property* 1PC-on-$G[S]$. *Let* $\lambda_{\mathcal{T}}(G[S])$ *be the size of* $\mathcal{P}_{\mathcal{T}}(G[S])$ *and let* v_n *be a non-terminal vertex. If during the process of vertex* v_n *the algorithm constructs a* 1PC $\mathcal{P}_{\mathcal{T}}(G[S-\{v_t\}])$ *of* $G[S-\{v_t\}]$ *of size* $\lambda_{\mathcal{T}}(G[S])-1$, *where* v_t *is the terminal vertex, and then connects the path* (v_t, v_n) *to an existing path* P, *then Algorithm* 1PC_Interval *computes a minimum* 1PC $\mathcal{P}_{\mathcal{T}}(G)$ *of the graph* G *satisfying Property* 1PC-on-G.

Proof Algorithm 1PC_INTERVAL computes a minimum 1PC $\mathcal{P}_{\mathcal{T}}(G[S])$ of the interval graph $G[S]$, $S = \{v_1, v_2, \ldots, v_{n-1}\}$, on at most $n-1$ vertices having $\varepsilon_i^{(n-1)}$ endpoints v_κ belonging to different paths with index $\kappa \in (i, n-1]$, $1 \le i \le n-1$, such that there is no other minimum 1PC $\mathcal{P}'_{\mathcal{T}}(G[S])$ having $\varepsilon_i'^{(n-1)}$ endpoints $v_{\kappa'}$ belonging to different paths with index $\kappa' \in (i, n-1]$ such that $\varepsilon_i'^{(n-1)} > \varepsilon_i^{(n-1)}$, $1 \le i < d$, where d is the index of the rightmost endpoint of a path in $\mathcal{P}_{\mathcal{T}}(G[S])$, and, moreover, exactly one of the following holds:

(i) $\varepsilon_i'^{(n-1)} \le \varepsilon_i^{(n-1)}$, $d \le i \le n-1$;
(ii) $\varepsilon_i'^{(n-1)} = \varepsilon_i^{(n-1)} + 1$, $d \le i < d'$ and $\varepsilon_i'^{(n-1)} = \varepsilon_i^{(n-1)}$, $d' \le i \le n-1$, where d' is the index of the rightmost endpoint of a path in $\mathcal{P}'_{\mathcal{T}}(G[S])$, and there exists a vertex v_q such that $\varepsilon_q^{(n-1)} > \varepsilon_q'^{(n-1)}$, and there exists no vertex $v_{q'}$ such that $\varepsilon_{q'}'^{(n-1)} > \varepsilon_{q'}^{(n-1)}$, $1 \le q, q' < d$.

When the algorithm processes vertex v_n, it constructs a 1PC $\mathcal{P}_{\mathcal{T}}(G[S-\{v_t\}])$ of $G[S-\{v_t\}]$ of size $\lambda_{\mathcal{T}}(G[S])-1$, where v_t is the terminal vertex, and then connects the path (v_t, v_n) to an existing path P. This operation is performed when vertex v_n sees the endpoints of at least one non-terminal path, say $P_1 = (v_r, \ldots, v_s)$, the terminal vertex v_t and no other endpoint of the terminal path, say, $P_2 = (v_t, \ldots, v_\ell)$. Then, the terminal path, P_2, has the same endpoints as in $\mathcal{P}_{\mathcal{T}}(G[S])$, the vertices of P_1 become internal vertices of P_2, while all the other paths in $\mathcal{P}_{\mathcal{T}}(G)$ remain the same as in $\mathcal{P}_{\mathcal{T}}(G[S])$. It is easy to see that there exists a path P in $\mathcal{P}_{\mathcal{T}}(G[S-\{v_t\}])$ having an endpoint, say, $v_{s'}$, with index greater than t, and thus, $v_{s'} \in N(v_n)$. Note that when the connect operation is performed, it may use a vertex of the terminal path in order to increase the value of an $\varepsilon_i^{(n-1)}$, $1 \le i < n-1$, and, in this case, $\varepsilon_d^{(n-1)} < \varepsilon_d'^{(n-1)}$. Then, for the endpoints of the terminal path, say, $v_t \in \mathcal{T}$ and v_ℓ, we have $v_t < v_\ell$. Consequently, since vertex v_n sees the endpoints of P_1, the terminal vertex v_t and no other endpoint of the terminal path P_2, if operation connect was called previously, it cannot have used a vertex of the terminal path and, thus, $\varepsilon_d^{(n)} < \varepsilon_d'^{(n)}$ cannot hold. Consequently, $\varepsilon_i'^{(n-1)} \le \varepsilon_i^{(n-1)}$, $d \le i \le n-1$.

The above procedure results to a 1PC of G of size $\lambda_{\mathcal{T}}(G) = \lambda_{\mathcal{T}}(G[S]) - 1$; consequently, $\mathcal{P}_{\mathcal{T}}(G)$ is a minimum 1PC of G, that is, $\lambda'_{\mathcal{T}}(G) = \lambda_{\mathcal{T}}(G)$.

We show that the algorithm computes a minimum 1PC $\mathcal{P}_{\mathcal{T}}(G)$ of the graph G having $\varepsilon_i^{(n)}$ endpoints v_κ belonging to different paths with index $\kappa \in (i, n]$, $1 \le i \le n$, such that there is no other minimum 1PC $\mathcal{P}'_{\mathcal{T}}(G)$ having $\varepsilon_i'^{(n)}$ endpoints $v_{\kappa'}$ belonging to different paths with index $\kappa' \in (i, n]$ such that $\varepsilon_i'^{(n)} > \varepsilon_i^{(n)}$, $1 \le i \le n$. Clearly, vertex v_n is an internal vertex of a path in any other minimum 1PC $\mathcal{P}'_{\mathcal{T}}(G)$, otherwise removing it from $\mathcal{P}'_{\mathcal{T}}(G)$ would result to a 1PC of $G[S]$ of size less or equal to $\lambda_{\mathcal{T}}(G[S]) - 1$, a contradiction. Suppose that there exists a minimum 1PC $\mathcal{P}'_{\mathcal{T}}(G)$ having an index, say, $k-1$, such that $\varepsilon'^{(n)}_{k-1} > \varepsilon^{(n)}_{k-1}$ and $1 \le k-1 < t-1$. This implies that $\varepsilon'^{(n)}_{k} = \varepsilon^{(n)}_{k}$ and there exists a vertex $v_{k'-1}$ such that $k'-1 < k-1$ and $\varepsilon'^{(n)}_{k'-1} = \varepsilon^{(n)}_{k'-1}$. Clearly, $v_{k'}v_n \notin E(G)$. Removing v_n from $\mathcal{P}'_{\mathcal{T}}(G)$ results to a minimum 1PC of $G[S]$ having at least two non-terminal neighbors of v_n as endpoints belonging to different paths, say, v_f and v_g; suppose that at least one of them has index greater than k. Then, $\varepsilon'^{(n-1)}_{k-1} > \varepsilon^{(n-1)}_{k-1}$, a contradiction. Thus, $v_{k'} < v_f < v_k$ and $v_{k'} < v_g < v_k$ and the right endpoint of the path that they belong, has also index less than k. Thus, $\varepsilon'^{(n-1)}_{k'} = \varepsilon'^{(n)}_{k'} + 1 = \varepsilon^{(n)}_{k'} + 1 + 1 = \varepsilon^{(n-1)}_{k'} + 1$ or $\varepsilon'^{(n-1)}_{k'} = \varepsilon'^{(n)}_{k'} + 2 = \varepsilon^{(n)}_{k'} + 1 + 2 = \varepsilon^{(n-1)}_{k'} + 2$, a contradiction. Suppose now that $t \le k-1 \le n$. Since there cannot exist a vertex v_ℓ such that $1 \le \ell < t-1$ and $\varepsilon'^{(n)}_{\ell} > \varepsilon^{(n)}_{\ell}$, there exists a vertex $v_{k'-1}$ such that $k'-1 < k-1$ and $\varepsilon'^{(n)}_{k'-1} = \varepsilon^{(n)}_{k'-1}$ and $t \le k'$. Clearly, $v_{k'}v_n \in E(G)$. If v_s is the right endpoint of P_1, then $k-1 < s$ and thus $k'-1 < s$. However, according to the algorithm, there cannot exist an endpoint between vertices v_t and v_s, a contradiction. □

Lemma 4.3 *Let G be an interval graph and suppose that Algorithm* 1PC_Interval *computes a minimum* 1PC $\mathcal{P}_{\mathcal{T}}(G[S])$ *of the graph* $G[S]$, $S = \{v_1, v_2, \dots, v_{n-1}\}$, *on at most $n-1$ vertices satisfying Property* 1PC-on-$G[S]$. *Let $\lambda_{\mathcal{T}}(G[S])$ be the size of $\mathcal{P}_{\mathcal{T}}(G[S])$ and let v_n be a non-terminal vertex. If during the process of vertex v_n the algorithm constructs a* 1PC $\mathcal{P}_{\mathcal{T}}(G[S - \{v_t\}])$ *of* $G[S - \{v_t\}]$ *of size* $\lambda_{\mathcal{T}}(G[S])$, *where v_t is the terminal vertex, connects v_t to the leftmost left endpoint it sees, and uses v_n to bridge two paths, then Algorithm* 1PC_Interval *computes a minimum* 1PC $\mathcal{P}_{\mathcal{T}}(G)$ *of the graph G satisfying Property* 1PC-on-G.

Proof Algorithm 1PC_INTERVAL computes a minimum 1PC $\mathcal{P}_{\mathcal{T}}(G[S])$ of the interval graph $G[S]$, $S = \{v_1, v_2, \dots, v_{n-1}\}$, on at most $n-1$ vertices having $\varepsilon^{(n-1)}_i$ endpoints v_κ belonging to different paths with index $\kappa \in (i, n-1]$, $1 \le i \le n-1$, such that there is no other minimum 1PC $\mathcal{P}'_{\mathcal{T}}(G[S])$ having $\varepsilon'^{(n-1)}_i$ endpoints $v_{\kappa'}$ belonging to different paths with index $\kappa' \in (i, n-1]$ such that $\varepsilon'^{(n-1)}_i > \varepsilon^{(n-1)}_i$, $1 \le i < d$, where d is the index of the rightmost endpoint of a path in $\mathcal{P}_{\mathcal{T}}(G[S])$, and, moreover, one of the following holds:

(i) $\varepsilon'^{(n-1)}_i \le \varepsilon^{(n-1)}_i$, $d \le i \le n-1$;
(ii) $\varepsilon'^{(n-1)}_i = \varepsilon^{(n-1)}_i + 1$, $d \le i < d'$ and $\varepsilon'^{(n-1)}_i = \varepsilon^{(n-1)}_i$, $d' \le i \le n-1$, where d' is the index of the rightmost endpoint of a path in $\mathcal{P}'_{\mathcal{T}}(G[S])$, and there exists a vertex v_q such that $\varepsilon^{(n-1)}_q > \varepsilon'^{(n-1)}_q$, and there exists no vertex $v_{q'}$ such that $\varepsilon'^{(n-1)}_{q'} > \varepsilon^{(n-1)}_{q'}$, $1 \le q, q' < d$.

When the algorithm processes vertex v_n, it constructs a 1PC $\mathcal{P}_{\mathcal{T}}(G[S - \{v_t\}])$ of $G[S - \{v_t\}]$ of size $\lambda_{\mathcal{T}}(G[S])$, where v_t is the terminal vertex, it connects v_t to the leftmost left endpoint it sees, and it uses v_n to bridge two paths. This operation is performed when vertex v_n sees the endpoints of at least one non-terminal path, say $P_1 =$

$(v_r, \ldots, v_s)$, the terminal vertex v_t of the terminal path, say, $P_2 = (v_t, v_j, \ldots, v_\ell)$ and an internal vertex v_j of P_2, and it does not see v_ℓ. Then, the terminal path, P_2, has the same endpoints as in $\mathcal{P}_\mathcal{T}(G[S])$, the vertices of P_1 become internal vertices of P_2, while all the other paths remain the same. Recall that, when the connect operation is performed, it may use a vertex of the terminal path in order to increase the value of an $\varepsilon_i^{(n-1)}$, $1 \le i < n-1$, and, in this case, $\varepsilon_d^{(n-1)} < \varepsilon_d'^{(n-1)}$. Then, for the endpoints of the terminal path, say, $v_t \in \mathcal{T}$ and v_ℓ, we have $v_t < v_\ell$. Consequently, since vertex v_n sees the endpoints of P_1, the terminal vertex v_t and not v_ℓ, if operation connect was called previously, it cannot have used a vertex of the terminal path and, thus, $\varepsilon_d^{(n-1)} < \varepsilon_d'^{(n-1)}$ cannot hold. Consequently, $\varepsilon_i'^{(n-1)} \le \varepsilon_i^{(n-1)}$, $d \le i \le n-1$.

Consider the case where vertex v_n sees the endpoints of only one non-terminal path, that is, of P_1. If applying the algorithm to $G[S - \{v_t\}]$ results to a 1PC of size $\lambda_\mathcal{T}(G[S]) - 1$ then it contains a path with one endpoint v_k such that $t < k$. Indeed, suppose that there does not exist such a path and P_1 consists of more than one vertex. This implies that all vertices of P_1 have bridged two paths and therefore we obtain a 1PC of size less than $\lambda_\mathcal{T}(G[S]) - 1$, a contradiction. Suppose now that the algorithm does not construct a path in the 1PC of $G[S - \{v_t\}]$ with one endpoint v_k such that $t < k$ and let the path P_1 consist of one vertex, say, $v_{k'}$. This implies that vertex $v_{k'}$ has bridged two paths and the same holds for every vertex v_i, $t \le i \le n-1$. Thus, if $v_{k'}$ is removed from $G[S - \{v_t\}]$, the algorithm would construct a minimum 1PC of size $\lambda_\mathcal{T}(G[S])$. Since the size of $\mathcal{P}_\mathcal{T}(G[S])$ constructed by the algorithm is $\lambda_\mathcal{T}(G[S])$, removing v_t and $v_{k'}$ results to a 1PC of size $\lambda_\mathcal{T}(G[S]) - 1$. Consequently, $v_{k'}$ cannot be used for bridging two paths in $\mathcal{P}_\mathcal{T}(G[S - \{v_t\}])$. This implies that the 1PC of $G[S - \{v_t\}]$ constructed by the algorithm contains a path with an endpoint v_k such that $t < k$. It is easy to see that if vertex v_n sees the endpoints of more than one non-terminal path, applying the algorithm to $G[S - \{v_t\}]$ results to a 1PC of size $\lambda_\mathcal{T}(G[S]) - 1$ having a path with one endpoint v_k such that $t < k$.

The above procedure results to a 1PC of G of size $\lambda_\mathcal{T}(G) = \lambda_\mathcal{T}(G[S]) - 1$; consequently, $\mathcal{P}_\mathcal{T}(G)$ is a minimum 1PC of G, that is, $\lambda'_\mathcal{T}(G) = \lambda_\mathcal{T}(G)$.

Using similar arguments as in the proof of Lemma 4.2, we show that the algorithm computes a minimum 1PC $\mathcal{P}_\mathcal{T}(G)$ of the graph G having $\varepsilon_i^{(n)}$ endpoints v_κ belonging to different paths with index $\kappa \in (i, n]$, $1 \le i \le n$, such that there is no other minimum 1PC $\mathcal{P}'_\mathcal{T}(G)$ having $\varepsilon_i'^{(n)}$ endpoints $v_{\kappa'}$ belonging to different paths with index $\kappa' \in (i, n]$ such that $\varepsilon_i'^{(n)} > \varepsilon_i^{(n)}$, $1 \le i \le n$. □

Lemma 4.4 *Let G be an interval graph and suppose that Algorithm* 1PC_Interval *computes a minimum* 1PC $\mathcal{P}_\mathcal{T}(G[S])$ *of the graph* $G[S]$, $S = \{v_1, v_2, \ldots, v_{n-1}\}$, *on at most* $n-1$ *vertices satisfying Property* 1PC-*on*-$G[S]$. *Let* $\lambda_\mathcal{T}(G[S])$ *be the size of* $\mathcal{P}_\mathcal{T}(G[S])$ *and let* v_n *be a non-terminal vertex. If during the process of vertex* v_n *the algorithm connects* v_n *to a path or inserts* v_n *into a path, then Algorithm* 1PC_Interval *computes a minimum* 1PC $\mathcal{P}_\mathcal{T}(G)$ *of the graph* G *satisfying Property* 1PC-on-G.

Proof Algorithm 1PC_INTERVAL computes a minimum 1PC $\mathcal{P}_\mathcal{T}(G[S])$ of the interval graph $G[S]$, $S = \{v_1, v_2, \ldots, v_{n-1}\}$, on at most $n-1$ vertices having $\varepsilon_i^{(n-1)}$

endpoints v_κ belonging to different paths with index $\kappa \in (i, n-1]$, $1 \le i \le n-1$, such that there is no other minimum 1PC $\mathcal{P}'_\mathcal{T}(G[S])$ having $\varepsilon_i'^{(n-1)}$ endpoints $v_{\kappa'}$ belonging to different paths with index $\kappa' \in (i, n-1]$ such that $\varepsilon_i'^{(n-1)} > \varepsilon_i^{(n-1)}$, $1 \le i < d$, where d is the index of the rightmost endpoint of a path in $\mathcal{P}_\mathcal{T}(G[S])$, and, moreover, exactly one of the following holds:

(i) $\varepsilon_i'^{(n-1)} \le \varepsilon_i^{(n-1)}$, $d \le i \le n-1$;
(ii) $\varepsilon_i'^{(n-1)} = \varepsilon_i^{(n-1)} + 1$, $d \le i < d'$ and $\varepsilon_i'^{(n-1)} = \varepsilon_i^{(n-1)}$, $d' \le i \le n-1$, where d' is the index of the rightmost endpoint of a path in $\mathcal{P}'_\mathcal{T}(G[S])$, and there exists a vertex v_q such that $\varepsilon_q^{(n-1)} > \varepsilon_q'^{(n-1)}$, and there exists no vertex $v_{q'}$ such that $\varepsilon_{q'}'^{(n-1)} > \varepsilon_{q'}^{(n-1)}$, $1 \le q, q' < d$.

We distinguish the following cases, according to the operation performed by the algorithm during the process of vertex v_n.

Case A When the algorithm processes vertex v_n, it connects v_n to a path, that is, $\lambda_\mathcal{T}(G) = \lambda_\mathcal{T}(G[S])$. Suppose that there exists a 1PC $\mathcal{P}'_\mathcal{T}(G)$ of size $\lambda_\mathcal{T}(G[S]) - 1$, that is, vertex v_n is an internal vertex of a path P in $\mathcal{P}'_\mathcal{T}(G)$. We distinguish the following cases:

(i) $P = (v_k, \ldots, v_r, v_n, v_s, \ldots, v_\ell)$. Removing v_n from P results to a minimum 1PC of $G[S]$ having two (non-terminal) neighbors of v_n as endpoints belonging to different paths. Since the algorithm does not use v_n to bridge two paths, the constructed minimum 1PC of $G[S]$ does not have two (non-terminal) neighbors of v_n as endpoints belonging to different paths. Consequently, there is a minimum 1PC of $G[S]$ for which there exists an index i such that $\varepsilon_i'^{(n)} > \varepsilon_i^{(n)}$, a contradiction.

(ii) $P = (v_t, v_n, \ldots, v_b)$, where v_t is the terminal vertex. Removing v_n and v_t from P results to a minimum 1PC of $G[S]$ of size $\lambda_\mathcal{T}(G[S]) - 1$, a contradiction. Indeed, since the algorithm does not use v_n to bridge two paths, removing v_t from $G[S]$ results to $\lambda_\mathcal{T}(G[S])$ paths.

Consequently, there does not exist a 1PC $\mathcal{P}'_\mathcal{T}(G)$ of size $\lambda_\mathcal{T}(G[S]) - 1$, and, thus, the 1PC constructed by the algorithm is minimum.

Case A.1. Suppose that $\varepsilon_i'^{(n-1)} \le \varepsilon_i^{(n-1)}$, $d \le i \le n-1$. We show that the algorithm computes a minimum 1PC $\mathcal{P}_\mathcal{T}(G)$ of the graph G having $\varepsilon_i^{(n)}$ endpoints v_κ belonging to different paths with index $\kappa \in (i, n]$, $1 \le i \le n$, such that there is no other minimum 1PC $\mathcal{P}'_\mathcal{T}(G)$ having $\varepsilon_i'^{(n)}$ endpoints $v_{\kappa'}$ belonging to different paths with index $\kappa' \in (i, n]$ such that $\varepsilon_i'^{(n)} > \varepsilon_i^{(n)}$, $1 \le i < \varrho$, and exactly one of the following holds:

(i) $\varepsilon_i'^{(n)} \le \varepsilon_i^{(n)}$, $\varrho \le i \le n$;
(ii) $\varepsilon_i'^{(n)} = \varepsilon_i^{(n)} + 1$, $\varrho \le i < \varrho'$ and $\varepsilon_i'^{(n)} = \varepsilon_i^{(n)}$, $\varrho' \le i \le n$, and there exists a vertex v_z such that $\varepsilon_z^{(n)} > \varepsilon_z'^{(n)}$, and there exists no vertex $v_{z'}$ such that $\varepsilon_{z'}'^{(n)} > \varepsilon_{z'}^{(n)}$, $1 \le z, z' < \varrho$.

Suppose that v_n is not an endpoint in $\mathcal{P}_\mathcal{T}(G)$; let $P = (\ldots, v_a, v_n, v_b, \ldots)$. According to operation connect, we break the terminal path of $\mathcal{P}_\mathcal{T}(G[S])$, which has the terminal vertex, v_t, as its right endpoint. Note that the terminal vertex is the second rightmost endpoint in $\mathcal{P}_{\mathcal{T}}(G[S])$ (see Sect. 3). The second rightmost endpoint of $\mathcal{P}_{\mathcal{T}}(G)$ has index greater than t, and v_t becomes a left endpoint of a path in $\mathcal{P}_{\mathcal{T}}(G)$. Assume that there exists a minimum 1PC $\mathcal{P}'_{\mathcal{T}}(G)$ having an index, say, $k-1$, for which we have $\varepsilon'^{(n)}_{k-1}$ endpoints $v_{\kappa'}$ belonging to different paths with index $\kappa' \in (k-1, n]$, where $\varepsilon'^{(n)}_{k-1} > \varepsilon^{(n)}_{k-1}$. Similarly, to Case 1 of Lemma 4.1, there exists a vertex v_j, $1 \le j < k-1$, such that $\varepsilon^{(n)}_j = \varepsilon'^{(n)}_j$.

Suppose that v_n is an endpoint of a path $P' = (v_n, v_{a'}, \ldots) \in \mathcal{P}'_{\mathcal{T}}(G)$. Then, since $\varrho' = n$, $\varepsilon'^{(n)}_i = \varepsilon^{(n)}_i + 1$, $\varrho \le i \le n$. Note that, there cannot exist a minimum 1PC $\mathcal{P}'_{\mathcal{T}}(G)$ such that $\varepsilon'^{(n)}_i = \varepsilon^{(n)}_i + 2$, $\varrho \le i \le n$. Indeed, suppose that there exists a minimum 1PC $\mathcal{P}'_{\mathcal{T}}(G)$ such that $\varepsilon'^{(n)}_i = \varepsilon^{(n)}_i + 2$, $\varrho \le i \le n$. If we remove v_n from $\mathcal{P}'_{\mathcal{T}}(G)$ we obtain $\varepsilon'^{(n-1)}_{\varrho} = \varepsilon'^{(n)}_{\varrho}$ or $\varepsilon'^{(n-1)}_{\varrho} = \varepsilon'^{(n)}_{\varrho} - 1$. However, $\varepsilon'^{(n)}_{\varrho} = \varepsilon^{(n)}_{\varrho} + 2$ and $\varepsilon^{(n)}_{\varrho} = \varepsilon^{(n-1)}_{\varrho} = 0$, a contradiction. According to the connect operation, $v_n v_{j+1} \notin E(G)$, thus $v_{a'} > v_{j+1}$. If we remove v_n from $\mathcal{P}'_{\mathcal{T}}(G)$ we obtain $\varepsilon'^{(n-1)}_{j+1} = \varepsilon'^{(n)}_{j+1} = \varepsilon^{(n)}_{j+1} + 1$. However, $\varepsilon^{(n)}_{j+1} = \varepsilon^{(n-1)}_{j+1}$ or $\varepsilon^{(n)}_{j+1} = \varepsilon^{(n-1)}_{j+1} + 1$, a contradiction. Consequently, there exists no vertex $v_{z'}$, $1 \le z' < \varrho$, such that $\varepsilon'^{(n)}_{z'} > \varepsilon^{(n)}_{z'}$. Suppose that $\varepsilon'^{(n)}_i = \varepsilon^{(n)}_i$, $1 \le i < \varrho$. Let v_r be the new endpoint created by the connect operation. Again, since $v_n v_{r-1} \notin E(G)$, $v_{a'} > v_{r-1}$ and if we remove v_n from $\mathcal{P}'_{\mathcal{T}}(G)$ we obtain $\varepsilon'^{(n-1)}_{r-1} = \varepsilon'^{(n)}_{r-1} = \varepsilon^{(n)}_{r-1}$. However, $\varepsilon^{(n)}_{r-1} = \varepsilon^{(n-1)}_{r-1} + 1$, a contradiction. Consequently, if $\varepsilon'^{(n)}_i = \varepsilon^{(n)}_i + 1$, $\varrho \le i < \varrho'$ and $\varepsilon'^{(n)}_i = \varepsilon^{(n)}_i$, $\varrho' \le i \le n$ then there exists a vertex v_z such that $\varepsilon^{(n)}_z > \varepsilon'^{(n)}_z$, $1 \le z < \varrho$ and there exists no vertex $v_{z'}$, $1 \le z' < \varrho$, such that $\varepsilon'^{(n)}_{z'} > \varepsilon^{(n)}_{z'}$. Suppose that the second rightmost endpoint in $\mathcal{P}_{\mathcal{T}}(G)$, say, v_f, has index less than the second rightmost endpoint in $\mathcal{P}'_{\mathcal{T}}(G)$, say, $v_{f'}$, that is, $v_f < v_{f'}$. Then, $\varepsilon^{(n-1)}_{f-1} = \varepsilon^{(n)}_{f-1} - 1 = \varepsilon'^{(n)}_{f-1} - 2$ and $\varepsilon'^{(n-1)}_{f-1} = \varepsilon'^{(n)}_{f-1} - 1$, a contradiction.

Suppose that v_n is not an endpoint in $\mathcal{P}'_{\mathcal{T}}(G)$; let $P' = (\ldots, v_{a'}, v_n, v_{b'}, \ldots)$. Clearly, one of $v_{a'}, v_{b'}$ is a vertex that could not be an endpoint in $\mathcal{P}'_{\mathcal{T}}(G)$. We show that $\varrho' \le \varrho$. Suppose that $\varrho' > \varrho$. Since $\varepsilon^{(n-1)}_{\varrho} = \varepsilon^{(n)}_{\varrho} = \varepsilon'^{(n)}_{\varrho} - 1 = 0$, then we have $\varepsilon'^{(n-1)}_{\varrho} = \varepsilon'^{(n)}_{\varrho} - 1$, which implies that the new endpoint in $\mathcal{P}'_{\mathcal{T}}(G)$ has index greater than the new endpoint created in $\mathcal{P}_{\mathcal{T}}(G)$, a contradiction. It is easy to see that there cannot exist a minimum 1PC $\mathcal{P}'_{\mathcal{T}}(G)$ having an index, say, $k-1$, for which we have $\varepsilon'^{(n)}_{k-1}$ endpoints $v_{\kappa'}$ belonging to different paths with index $\kappa' \in (k-1, n]$ such that $\varepsilon'^{(n)}_{k-1} = \varepsilon'^{(n-1)}_{k-1} + 2$. Let v_t be the terminal vertex. We have $\varepsilon^{(n)}_{t-1} = \varepsilon'^{(n-1)}_{t-1}$. It is easy to see that there cannot exist a minimum 1PC $\mathcal{P}'_{\mathcal{T}}(G)$ having an index, say, $k-1$, $k-1 \le t-1$, for which we have $\varepsilon'^{(n)}_{k-1}$ endpoints $v_{\kappa'}$ belonging to different paths with index $\kappa' \in (k-1, n]$, such that $\varepsilon'^{(n)}_{k-1} > \varepsilon^{(n)}_{k-1}$.

Suppose that v_n is the right endpoint of a path $P = (v_n, v_a, \ldots) \in \mathcal{P}_{\mathcal{T}}(G)$. Then, there cannot exist a minimum 1PC $\mathcal{P}'_{\mathcal{T}}(G)$ such that $1 = \varepsilon^{(n)}_{\varrho-1} < \varepsilon'^{(n)}_{\varrho-1}$. We show that the algorithm computes a minimum 1PC $\mathcal{P}_{\mathcal{T}}(G)$ of the graph G having $\varepsilon^{(n)}_i$ endpoints v_κ belonging to different paths with index $\kappa \in (i, n]$, $1 \le i \le n$, such that there is no other minimum 1PC $\mathcal{P}'_{\mathcal{T}}(G)$ having $\varepsilon_i^{\prime(n)}$ endpoints $v_{\kappa'}$ belonging to different paths with index $\kappa' \in (i, n]$ such that $\varepsilon_i^{\prime(n)} > \varepsilon_i^{(n)}$, $1 \le i \le n$. Assume that there exists a minimum 1PC $\mathcal{P}'_{\mathcal{T}}(G)$ having an index, say, $k-1$, for which we have $\varepsilon_{k-1}^{\prime(n)}$ endpoints $v_{\kappa'}$ belonging to different paths with index $\kappa' \in (k-1, n]$, where $\varepsilon_{k-1}^{\prime(n)} > \varepsilon_{k-1}^{(n)}$. Similarly, to Case 1 of Lemma 4.1, there exists a vertex v_j, $1 \le j < k-1$, such that $\varepsilon_j^{(n)} = \varepsilon_j^{\prime(n)}$.

Suppose that v_n is an endpoint of a path $P' = (v_n, v_{a'}, \dots) \in \mathcal{P}'_{\mathcal{T}}(G)$. Note that if $v_{a'} < v_{j+1}$, then $v_{j+1}v_n \in E(G)$ and v_k should be the terminal vertex belonging to a non-trivial path, otherwise $\mathcal{P}_{\mathcal{T}}(G)$ would not be minimum. Thus, if v_{j+1} is not the terminal vertex, $v_a < v_{j+1}$ or $v_a = v_{j+1}$. Then, if we remove v_n from $\mathcal{P}'_{\mathcal{T}}(G)$ and $\mathcal{P}_{\mathcal{T}}(G)$, we obtain $\varepsilon_{j+1}^{(n-1)} = \varepsilon_{j+1}^{(n)} - 1 = \varepsilon_{j+1}^{\prime(n)} - 2$ and $\varepsilon_{j+1}^{\prime(n-1)} = \varepsilon_{j+1}^{\prime(n)} - 1$; thus, $\varepsilon_{j+1}^{(n-1)} = \varepsilon_{j+1}^{\prime(n-1)} - 1$, a contradiction. On the other hand, if $v_{a'} > v_{j+1}$, then, if we remove v_n from $\mathcal{P}_{\mathcal{T}}(G)$ and $\mathcal{P}'_{\mathcal{T}}(G)$, we obtain $\varepsilon_{j+1}^{(n-1)} = \varepsilon_{j+1}^{(n)} - 1 = \varepsilon_{j+1}^{\prime(n)} - 2$ or $\varepsilon_{j+1}^{(n-1)} = \varepsilon_{j+1}^{(n)} = \varepsilon_{j+1}^{\prime(n)} - 1$. Also, $\varepsilon_{j+1}^{\prime(n-1)} = \varepsilon_{j+1}^{\prime(n)}$; thus, $\varepsilon_{j+1}^{(n-1)} = \varepsilon_{j+1}^{\prime(n-1)} - 1$, a contradiction.

Suppose that v_n is not an endpoint in $\mathcal{P}'_{\mathcal{T}}(G)$; let $P' = (\dots, v_{a'}, v_n, v_{b'}, \dots)$. If $v_{a'}v_{b'} \in E(G)$ then if we remove v_n from $\mathcal{P}'_{\mathcal{T}}(G)$ and $\mathcal{P}'_{\mathcal{T}}(G)$, we obtain $\varepsilon_{j+1}^{(n-1)} = \varepsilon_{j+1}^{(n)} - 1 = \varepsilon_{j+1}^{\prime(n)} - 2$ or $\varepsilon_{j+1}^{(n-1)} = \varepsilon_{j+1}^{(n)} = \varepsilon_{j+1}^{\prime(n)} - 1$. Also, $\varepsilon_{j+1}^{\prime(n-1)} = \varepsilon_{j+1}^{\prime(n)}$; thus, $\varepsilon_{j+1}^{(n-1)} = \varepsilon_{j+1}^{\prime(n-1)} - 1$, a contradiction. Consequently, $v_{a'}v_{b'} \notin E(G)$; however, we have shown that v_n becomes an endpoint in $\mathcal{P}_{\mathcal{T}}(G)$ only when a new right endpoint cannot be created by making v_n an internal vertex, a contradiction.

Case A.2. Suppose that $\varepsilon_i^{\prime(n-1)} = \varepsilon_i^{(n-1)} + 1$, $d \le i < d'$ and $\varepsilon_i^{\prime(n-1)} = \varepsilon_i^{(n-1)}$, $d' \le i \le n-1$ and there exists a vertex v_q such that $\varepsilon_q^{(n-1)} > \varepsilon_q^{\prime(n-1)}$, $1 \le q < d$ and there exists no vertex $v_{q'}$, $1 \le q' < d$, such that $\varepsilon_{q'}^{\prime(n-1)} > \varepsilon_{q'}^{(n-1)}$. We show that the algorithm computes a minimum 1PC $\mathcal{P}_{\mathcal{T}}(G)$ of the graph G having $\varepsilon_i^{(n)}$ endpoints v_κ belonging to different paths with index $\kappa \in (i, n]$, $1 \le i \le n$, such that there is no other minimum 1PC $\mathcal{P}'_{\mathcal{T}}(G)$ having $\varepsilon_i^{\prime(n)}$ endpoints $v_{\kappa'}$ belonging to different paths with index $\kappa' \in (i, n]$ such that $\varepsilon_i^{\prime(n)} > \varepsilon_i^{(n)}$, $1 \le i \le n$.

Assume that there exists a minimum 1PC $\mathcal{P}'_{\mathcal{T}}(G)$ having an index, say, $k-1$, for which we have $\varepsilon_{k-1}^{\prime(n)}$ endpoints $v_{\kappa'}$ belonging to different paths with index $\kappa' \in (k-1, n]$, where $\varepsilon_{k-1}^{\prime(n)} > \varepsilon_{k-1}^{(n)}$. Similarly, to Case 1 of Lemma 4.1, there exists a vertex v_j, $1 \le j < k-1$, such that $\varepsilon_j^{(n)} = \varepsilon_j^{\prime(n)}$. Using similar arguments as in Case A.1, we show that v_n is an endpoint of a path $P' = (v_n, v_{a'}, \dots) \in \mathcal{P}'_{\mathcal{T}}(G)$ and that there cannot exist a minimum 1PC $\mathcal{P}'_{\mathcal{T}}(G)$ having an index, say, $k-1$, for which we have $\varepsilon_{k-1}^{\prime(n)}$ endpoints $v_{\kappa'}$ belonging to different paths with index $\kappa' \in (k-1, n]$, where $\varepsilon_{k-1}^{\prime(n)} > \varepsilon_{k-1}^{(n)}$.

Case B When the algorithm processes vertex v_n, it inserts v_n into a path, that is, $\lambda_{\mathcal{T}}(G) = \lambda_{\mathcal{T}}(G[S])$. This implies that $\forall i \ge d$ we have $\varepsilon_i^{(n-1)} \le 1$. Suppose that there exists a 1PC $\mathcal{P}'_{\mathcal{T}}(G)$ of size $\lambda_{\mathcal{T}}(G[S]) - 1$, that is, vertex v_n is an internal vertex of a path P in $\mathcal{P}'_{\mathcal{T}}(G)$. Then, removing vertex v_n from $\mathcal{P}'_{\mathcal{T}}(G)$ results to a 1PC of $G[S]$ of size $\lambda_{\mathcal{T}}(G[S])$, and, thus, minimum, such that there exists an index i, $i \geq d$, for which $\varepsilon_i'^{(n-1)} = 2$, a contradiction.

Consequently, there does not exist a 1PC $\mathcal{P}'_{\mathcal{T}}(G)$ of size $\lambda_{\mathcal{T}}(G[S]) - 1$, and, thus, the 1PC constructed by the algorithm is minimum.

Case B.1. Suppose that $\varepsilon_i'^{(n-1)} \leq \varepsilon_i^{(n-1)}$, $d \leq i \leq n-1$. We show that the algorithm computes a minimum 1PC $\mathcal{P}_{\mathcal{T}}(G)$ of the graph G having $\varepsilon_i^{(n)}$ endpoints v_κ belonging to different paths with index $\kappa \in (i, n]$, $1 \leq i \leq n$, such that there is no other minimum 1PC $\mathcal{P}'_{\mathcal{T}}(G)$ having $\varepsilon_i'^{(n)}$ endpoints $v_{\kappa'}$ belonging to different paths with index $\kappa' \in (i, n]$ such that $\varepsilon_i'^{(n)} > \varepsilon_i^{(n)}$, $1 \leq i \leq n$.

Assume that there exists a minimum 1PC $\mathcal{P}'_{\mathcal{T}}(G)$ having an index, say, $k-1$, for which we have $\varepsilon_{k-1}'^{(n)}$ endpoints $v_{\kappa'}$ belonging to different paths with index $\kappa' \in (k-1, n]$, where $\varepsilon_{k-1}'^{(n)} > \varepsilon_{k-1}^{(n)}$. Similarly, to Case 1 of Lemma 4.1, there exists a vertex v_j, $1 \leq j < k-1$, such that $\varepsilon_j^{(n)} = \varepsilon_j'^{(n)}$.

Suppose that v_n is an endpoint of a path $P' = (v_n, v_t) \in \mathcal{P}'_{\mathcal{T}}(G)$, such that $v_t \in \mathcal{T}$. Then, $v_t v_n \in E(G)$ and the size of a minimum 1PC of $G[S - \{v_t\}]$ is $\lambda_{\mathcal{T}}(G) - 1$, a contradiction.

Suppose that v_n is an endpoint of a path $P' = (v_n, v_{a'}, \ldots, v_{b'}) \in \mathcal{P}'_{\mathcal{T}}(G)$, such that $v_{a'} \notin \mathcal{T}$. Then, removing vertex v_n from $\mathcal{P}'_{\mathcal{T}}(G)$ results to a 1PC of $G[S]$ of size $\lambda_{\mathcal{T}}(G[S])$, and, thus, minimum, such that there exists an index i for which $\varepsilon_i'^{(n-1)} = 1$, $i \geq d$. Then, $d = d'$, which is equal to the index of the terminal vertex, and P' is the terminal path such that its left endpoint in $\mathcal{P}'_{\mathcal{T}}(G[S])$, that is, vertex $v_{a'}$, has index greater than the index of the left endpoint of the terminal path in $\mathcal{P}_{\mathcal{T}}(G[S])$. Note that, $v_{a'}$ cannot be an endpoint in $\mathcal{P}'_{\mathcal{T}}(G[S])$ since $\varepsilon_i'^{(n-1)} \leq \varepsilon_i^{(n-1)}$, $d \leq i \leq n-1$, and, thus, a 1PC of $G[S]$ having $v_{a'}$ as an endpoint cannot be minimum. However, removing v_n from $\mathcal{P}'_{\mathcal{T}}(G)$ results to a 1PC of $G[S]$ of size $\lambda_{\mathcal{T}}(G[S])$ having $v_{a'}$ as an endpoint, a contradiction.

Suppose now that v_n is not an endpoint in $\mathcal{P}'_{\mathcal{T}}(G)$; let $P' = (\ldots, v_{a'}, v_n, v_{b'}, \ldots)$. If $v_{a'} v_{b'} \in E(G)$ then if we remove v_n from $\mathcal{P}'_{\mathcal{T}}(G)$ and $\mathcal{P}'_{\mathcal{T}}(G)$, we obtain $\varepsilon_{k-1}^{(n-1)} = \varepsilon_{k-1}^{(n)} = \varepsilon_{k-1}'^{(n)} - 1$ and $\varepsilon_{k-1}'^{(n-1)} = \varepsilon_{k-1}'^{(n)}$, a contradiction. Consequently, $v_{a'} v_{b'} \notin E(G)$. Suppose that the value of d-connectivity of $\mathcal{P}_{\mathcal{T}}(G[S])$ is c; then the value of d-connectivity of $\mathcal{P}_{\mathcal{T}}(G)$ is $c+2$. However, the corresponding value of $\mathcal{P}'_{\mathcal{T}}(G)$ is not increased by vertices $v_{a'}, v_n$ and $v_{b'}$, since $v_{a'}$ and $v_{b'}$ are internal vertices not successive into a path in a 1PC of $G[S]$ and there exist two vertices connected to $v_{a'}$ and $v_{b'}$ in $\mathcal{P}_{\mathcal{T}}(G[S])$, say, v_f and v_g, respectively, for which $d(v_f)$ and $d(v_g)$ are reduced, and, thus, they reduce the d-connectivity by two. In order to obtain $c+2$ for $\mathcal{P}'_{\mathcal{T}}(G)$ the vertices of $V(G) - \{v_{a'}, v_{b'}, v_n\}$ must increase the d-connectivity by two. However, the size of $\mathcal{P}'_{\mathcal{T}}(G)$ is also $\lambda_{\mathcal{T}}(G)$ and vertices $v_{a'}, v_{b'}$ and v_n are also internal in $\mathcal{P}_{\mathcal{T}}(G)$. Thus, increasing the d-connectivity by two is not possible and we have a contradiction. Note that $v_f v_g \notin E(G)$; otherwise we would have also $v_n v_f \in E(G)$ and $v_n v_g \in E(G)$ and there would exist a 1PC having the same endpoints as $\mathcal{P}'_{\mathcal{T}}(G)$ and containing a path $P = (\ldots, v_f, v_n, v_g, \ldots)$ with $v_f v_g \in E(G)$, a contradiction.

Case B.2. Suppose that $\varepsilon_i'^{(n-1)} = \varepsilon_i^{(n-1)} + 1$, $d \leq i < d'$ and $\varepsilon_i'^{(n-1)} = \varepsilon_i^{(n-1)}$, $d' \leq i \leq n-1$ and there exists a vertex v_q such that $\varepsilon_q^{(n-1)} > \varepsilon_q'^{(n-1)}$, $1 \leq q < d$

and there exists no vertex $v_{q'}$, $1 \le q' < d$, such that $\varepsilon_{q'}^{\prime(n-1)} > \varepsilon_{q'}^{(n-1)}$. Using similar arguments as in Case A.1 where v_n is not an endpoint in $\mathcal{P}_{\mathcal{T}}(G[S])$, we show that the algorithm computes a minimum 1PC $\mathcal{P}_{\mathcal{T}}(G)$ of the graph G having $\varepsilon_i^{(n)}$ endpoints v_κ belonging to different paths with index $\kappa \in (i, n]$, $1 \le i \le n$, such that there is no other minimum 1PC $\mathcal{P}'_{\mathcal{T}}(G)$ having $\varepsilon_i^{\prime(n)}$ endpoints $v_{\kappa'}$ belonging to different paths with index $\kappa' \in (i, n]$ such that $\varepsilon_i^{\prime(n)} > \varepsilon_i^{(n)}$, $1 \le i < \varrho$. Furthermore, if $\varepsilon_i^{\prime(n)} = \varepsilon_i^{(n)} + 1$, $\varrho \le i < \varrho'$ and $\varepsilon_i^{\prime(n)} = \varepsilon_i^{(n)}$, $\varrho' \le i \le n$ then there exists a vertex v_z such that $\varepsilon_z^{(n)} > \varepsilon_z^{\prime(n)}$, $1 \le z < \varrho$ and there exists no vertex $v_{z'}$, $1 \le z' < \varrho$, such that $\varepsilon_{z'}^{\prime(n)} > \varepsilon_{z'}^{(n)}$. □

□

Lemma 4.5 *Let G be an interval graph and suppose that Algorithm* 1PC_Interval *computes a minimum* 1PC $\mathcal{P}_{\mathcal{T}}(G[S])$ *of the graph* $G[S]$, $S = \{v_1, v_2, \ldots, v_{n-1}\}$, *on at most* $n-1$ *vertices satisfying Property* 1PC-on-$G[S]$. *Let* $\lambda_{\mathcal{T}}(G[S])$ *be the size of* $\mathcal{P}_{\mathcal{T}}(G[S])$ *and let* v_n *be a non-terminal vertex. If during the process of vertex* v_n *the algorithm creates a new path having vertex* v_n *as an endpoint then Algorithm* 1PC_Interval *computes a minimum* 1PC $\mathcal{P}_{\mathcal{T}}(G)$ *of the graph* G *satisfying Property* 1PC-on-G.

Proof Algorithm 1PC_INTERVAL computes a minimum 1PC $\mathcal{P}_{\mathcal{T}}(G[S])$ of the interval graph $G[S]$, $S = \{v_1, v_2, \ldots, v_{n-1}\}$, on at most $n-1$ vertices having $\varepsilon_i^{(n-1)}$ endpoints v_κ belonging to different paths with index $\kappa \in (i, n-1]$, $1 \le i \le n-1$, such that there is no other minimum 1PC $\mathcal{P}'_{\mathcal{T}}(G[S])$ having $\varepsilon_i^{\prime(n-1)}$ endpoints $v_{\kappa'}$ belonging to different paths with index $\kappa' \in (i, n-1]$ such that $\varepsilon_i^{\prime(n-1)} > \varepsilon_i^{(n-1)}$, $1 \le i < d$, where d is the index of the rightmost endpoint of a path in $\mathcal{P}_{\mathcal{T}}(G[S])$, and, moreover, exactly one of the following holds:

(i) $\varepsilon_i^{\prime(n-1)} \le \varepsilon_i^{(n-1)}$, $d \le i \le n-1$;
(ii) $\varepsilon_i^{\prime(n-1)} = \varepsilon_i^{(n-1)} + 1$, $d \le i < d'$ and $\varepsilon_i^{\prime(n-1)} = \varepsilon_i^{(n-1)}$, $d' \le i \le n-1$, where d' is the index of the rightmost endpoint of a path in $\mathcal{P}'_{\mathcal{T}}(G[S])$, and there exists a vertex v_q such that $\varepsilon_q^{(n-1)} > \varepsilon_q^{\prime(n-1)}$, and there exists no vertex $v_{q'}$ such that $\varepsilon_{q'}^{\prime(n-1)} > \varepsilon_{q'}^{(n-1)}$, $1 \le q, q' < d$.

When the algorithm processes vertex v_n, it creates a new path having vertex v_n as an endpoint, that is, $\lambda_{\mathcal{T}}(G) = \lambda_{\mathcal{T}}(G[S]) + 1$. This implies that $\forall i \ge d$ we have $\varepsilon_i^{(n-1)} \le 1$. Suppose that there exists a 1PC $\mathcal{P}'_{\mathcal{T}}(G)$ of size $\lambda_{\mathcal{T}}(G[S]) - 1$, that is, vertex v_n is an internal vertex of a path P in $\mathcal{P}'_{\mathcal{T}}(G)$. Then, removing vertex v_n from $\mathcal{P}'_{\mathcal{T}}(G)$ results to a 1PC of $G[S]$ of size $\lambda_{\mathcal{T}}(G[S])$, and, thus, minimum, such that there exists an index i for which $\varepsilon_i^{\prime(n-1)} = 2$, a contradiction.

Suppose now that there exists a 1PC $\mathcal{P}'_{\mathcal{T}}(G)$ of size $\lambda_{\mathcal{T}}(G[S])$. Let v_n be an endpoint of a path P in $\mathcal{P}'_{\mathcal{T}}(G)$. We distinguish the following cases:

(i) $P = (v_n, v_r, \ldots, v_s)$. Removing vertex v_n from $\mathcal{P}'_{\mathcal{T}}(G)$ results to a 1PC of $G[S]$ of size $\lambda_{\mathcal{T}}(G[S])$, and, thus, minimum, such that there exists an index i for which $\varepsilon_i^{\prime(n-1)} = 1$, $i \ge d$. Then, $d = d'$, which is equal to the index of the terminal vertex, and P is the terminal path such that its left endpoint in $\mathcal{P}'_{\mathcal{T}}(G[S])$, that is, vertex v_r, has index greater than the index of the left endpoint of the terminal path in

$\mathcal{P}_{\mathcal{T}}(G[S])$. Note that, v_r cannot be an endpoint in $\mathcal{P}'_{\mathcal{T}}(G[S])$ since $\varepsilon_i'^{(n-1)} \leq \varepsilon_i^{(n-1)}$, $d \leq i \leq n-1$, and, thus, a 1PC of $G[S]$ having v_r as an endpoint cannot be minimum. However, removing v_n from $\mathcal{P}'_{\mathcal{T}}(G)$ results to a 1PC of $G[S]$ of size $\lambda_{\mathcal{T}}(G[S])$ having v_r as an endpoint, a contradiction.

(ii) $P = (v_t, v_n)$, where v_t is the terminal vertex. Removing v_n and v_t from P results to a 1PC of $G[S - \{v_t\}]$ of size $\lambda_{\mathcal{T}}(G[S]) - 1$, a contradiction. Indeed, since the algorithm does not use v_n to bridge two paths, removing v_t from $G[S]$ results to $\lambda_{\mathcal{T}}(G[S])$ paths.

Now let v_n be an internal vertex of a path $P = (v_r, \ldots, v_i, v_n, v_j, \ldots, v_s)$ in $\mathcal{P}'_{\mathcal{T}}(G)$. Suppose that $N(v_n) > 0$ (the case where $N(v_n) = 0$ is trivial) and $v_t \notin N(v_n)$, where v_t is the terminal vertex. Since the algorithm constructs $\lambda_{\mathcal{T}}(G[S]) + 1$ paths, at least $|N(v_n)| - 1$ neighbors of v_n have bridged paths reducing the size of the 1PC and at most one of them was inserted; otherwise there would exist at least two successive neighbors into a path or at least one of them would be an endpoint. Suppose that v_i and v_j have both bridged paths. Then, applying the algorithm to $G - \{v_i, v_j, v_n\}$ would result to a minimum 1PC of $G - \{v_i, v_j, v_n\}$ of size $\lambda_{\mathcal{T}}(G[S]) + 2$. However, if we remove vertices v_i, v_j and v_n from $\mathcal{P}'_{\mathcal{T}}(G)$ we obtain a 1PC of $G - \{v_i, v_j, v_n\}$ of size $\lambda_{\mathcal{T}}(G[S]) + 1$, a contradiction. Suppose now that v_i was inserted and v_j has bridged paths. We distinguish the following cases:

(i) $j < i$. Clearly, applying the algorithm to $G' = G - \{v_i, v_n\}$ results to a minimum 1PC of G' of size $\lambda_{\mathcal{T}}(G[S])$. Furthermore, applying the algorithm to $G' - \{v_j\}$ results to a minimum 1PC $\mathcal{P}''_1(G)$ of $G' - \{v_j\}$ of size $\lambda_{\mathcal{T}}(G[S]) + 1$ such that no non-terminal neighbor of v_i is an endpoint and if v_i sees the terminal vertex v_t, it is not a trivial path in $\mathcal{P}''_1(G)$. Indeed, any neighbor v_a of v_i such that $i < a < n$ cannot be an endpoint in $\mathcal{P}''_1(G)$ since every vertex v_a such that $i < a < n$ is also a neighbor of v_n. Note that, $t < j$. Furthermore, since v_i is inserted, when vertex v_{j+1} was processed, no neighbor of v_i was an endpoint and if $v_i v_t \in E(G)$ vertex v_t does not belong to a trivial path. Indeed, let $v_k \in N(v_i)$ be an endpoint when the algorithm processes vertex v_{j+1} or v_t belongs to a trivial path. This implies that, when we apply the algorithm to $G[S]$, we have one neighbor of v_n, say, v_ℓ, bridging through vertex v_k or vertex v_t; then v_i would be inserted through the edge $v_k v_\ell$ or $v_t v_\ell$, which is a contradiction since this results to two neighbors of v_n being successive. Additionally, no neighbor of v_i becomes an endpoint and vertex v_t does not belong to a trivial path until vertex v_{n-1} is processed, since all vertices with index greater than $j+1$ are neighbors of v_n, and, thus, they are used to bridge paths reducing the size of the 1PC. Note that, according to the algorithm, vertex v_t cannot belong to a trivial path until vertex v_{n-1} is processed, since no bridge operation results to v_t belonging to a trivial path. Consequently, applying the algorithm to $G' - \{v_j\}$ results to a minimum 1PC $\mathcal{P}''_1(G' - \{v_j\})$ of size $\lambda_{\mathcal{T}}(G[S]) + 1$ such that no non-terminal neighbor of v_i is an endpoint and if v_i sees the terminal vertex v_t, it is not a trivial path in $\mathcal{P}''_1(G' - \{v_j\})$.

(ii) $i < j$. Similarly to case (i), applying the algorithm to $G' - \{v_j\}$ results to a minimum 1PC $\mathcal{P}''_1(G' - \{v_j\})$ of size $\lambda_{\mathcal{T}}(G[S]) + 1$ such that no non-terminal neighbor of v_i is an endpoint and if v_i sees the terminal vertex v_t, it is not a trivial path in $\mathcal{P}''_1(G' - \{v_j\})$. Indeed, when vertex v_{i+1} is processed, no neighbor of v_i is an endpoint and if v_i sees the terminal vertex v_t, it is not a trivial path. Furthermore, since no neighbor of v_n can be an endpoint, no vertex with index greater than $i+1$ is an endpoint. Additionally, no neighbor of v_i becomes an endpoint and if $v_t v_i \in E(G)$, vertex v_t does not belong to a trivial path until vertex v_{n-1} is processed, since all vertices with index greater than $i+1$ are neighbors of v_n, and, thus, they are used to bridge paths reducing the size of the 1PC. Note that, according to the algorithm, vertex v_t cannot belong to a trivial path until vertex v_{n-1} is processed. Consequently, applying the algorithm to $G' - \{v_j\}$ results to a minimum 1PC $\mathcal{P}''{}_1(G' - \{v_j\})$ of size $\lambda_{\mathcal{T}}(G[S]) + 1$ such that no non-terminal neighbor of v_i is an endpoint and if v_i sees the terminal vertex v_t, it is not a trivial path in $\mathcal{P}''{}_1(G' - \{v_j\})$.

Since v_n is an internal vertex of a path $P = (v_r, \dots, v_i, v_n, v_j, \dots, v_s)$ in $\mathcal{P}'_{\mathcal{T}}(G)$ which has size $\lambda_{\mathcal{T}}(G[S])$, if we remove vertices v_i, v_j and v_n from P we obtain a 1PC of $G' - \{v_j\}$ of size $\lambda_{\mathcal{T}}(G[S]) + 1$ such that a non-terminal neighbor of v_i is an endpoint, a contradiction; the same holds when $P = (v_r, \dots, v_i, v_n, v_j, v_t)$. If $P = (v_t, v_i, v_n, v_j, \dots, v_s)$ then v_t belongs to a trivial path in $\mathcal{P}''{}_1(G)$, a contradiction. If $P = (v_i, v_n, v_j, \dots, v_s)$ or $P = (v_r, \dots, v_i, v_n, v_j)$, then removing vertices v_i, v_j and v_n from P results to a 1PC of $G' - \{v_j\}$ of size $\lambda_{\mathcal{T}}(G[S])$, a contradiction.

Now let $v_t \in N(v_n)$, where v_t is the terminal vertex. The case where v_n is an internal vertex of a path $P = (v_r, \dots, v_i, v_n, v_j, \dots, v_s)$ in $\mathcal{P}'_{\mathcal{T}}(G)$ which has size $\lambda_{\mathcal{T}}(G[S])$ leads to a contradiction similarly to the case where $v_t \notin N(v_n)$. Suppose that v_n is an internal vertex of a path $P = (v_t, v_n, v_j, \dots, v_s)$ in $\mathcal{P}'_{\mathcal{T}}(G)$ which has size $\lambda_{\mathcal{T}}(G[S])$. According to the algorithm, no neighbor of v_n is inserted until vertex v_t is processed. Also, it is easy to see that, no neighbor of v_n with index greater than t is inserted, either. Indeed, let v_a, $t < a < n$, be a neighbor of v_n which is inserted into the terminal path. Since the algorithm results to $\lambda_{\mathcal{T}}(G[S]) + 1$ paths, v_a is inserted through an edge $v_k v_\ell$ such that $v_k, v_\ell \notin N(v_n)$ and v_t is connected to a vertex v_q such that $v_q \notin N(v_n)$. This implies that the ordering of the vertices v_t, v_k, v_ℓ and v_q of the terminal path is as follows: $v_q < v_k < v_\ell < v_t$ or $v_q < v_\ell < v_k < v_t$. Without loss of generality suppose that $v_q < v_k < v_\ell < v_t$. Let v_b be the other endpoint of the terminal path. Clearly $v_b < v_k$. Also, without loss of generality, suppose that $v_b < v_q$. Consequently, when vertex v_t is processed, the algorithm has constructed a path having two successive vertices, v_k and v_ℓ, which have indexes greater than those of the endpoints of the path, that is, v_b and v_q. This is a contradiction, since it implies that there exists at least one vertex with index greater than q which sees v_q; in this case the algorithm could not result to a path having v_q as an endpoint. Consequently, we have shown that if we apply the algorithm to $G[S]$, no neighbor of v_n is inserted into the terminal path. Furthermore, since there are no neighbors of v_n successive into a path, all neighbors of v_n bridge paths reducing the size of the 1PC. This implies that, if we apply the algorithm to $G[S - \{v_t\}]$, we obtain a minimum 1PC of $G[S - \{v_t\}]$ of size $\lambda_{\mathcal{T}}(G[S])$. Furthermore, if we apply the algorithm to $G[S - \{v_t, v_j\}]$, we obtain a minimum 1PC of $G[S - \{v_t, v_j\}]$ of size $\lambda_{\mathcal{T}}(G[S]) + 1$. However, removing v_t, v_n and v_j from $P = (v_t, v_n, v_j, \dots, v_s)$ which is a path in $\mathcal{P}'_{\mathcal{T}}(G)$, we obtain a 1PC of $G[S - \{v_t, v_j\}]$ of size at most $\lambda_{\mathcal{T}}(G[S])$, a contradiction; thus, v_n cannot be an internal vertex of a path $P = (v_t, v_n, v_j, \dots, v_s)$ in $\mathcal{P}'_{\mathcal{T}}(G)$ which has size $\lambda_{\mathcal{T}}(G[S])$.

We have shown that there does not exist a 1PC $\mathcal{P}'_{\mathcal{T}}(G)$ of size $\lambda_{\mathcal{T}}(G[S])$, and, thus, $\mathcal{P}_{\mathcal{T}}(G)$ is a minimum 1PC of G, that is, $\lambda'_{\mathcal{T}}(G) = \lambda_{\mathcal{T}}(G) = \lambda_{\mathcal{T}}(G[S]) + 1$.

Using similar arguments as in Case A.1 where v_n is an endpoint in $\mathcal{P}_{\mathcal{T}}(G[S])$, we show that the algorithm computes a minimum 1PC $\mathcal{P}_{\mathcal{T}}(G)$ of every interval graph G with n vertices having $\varepsilon_i^{(n)}$ endpoints v_κ belonging to different paths with index $\kappa \in (i, n]$, $1 \le i \le n$, such that there is no other minimum 1PC $\mathcal{P}'_{\mathcal{T}}(G)$ having $\varepsilon_i^{\prime(n)}$ endpoints $v_{\kappa'}$ belonging to different paths with index $\kappa' \in (i, n]$ such that $\varepsilon_i^{\prime(n)} > \varepsilon_i^{(n)}$, $1 \le i \le n$. □

Lemma 4.6 *Let G be an interval graph and suppose that Algorithm* 1PC_INTERVAL *computes a minimum* 1PC $\mathcal{P}_{\mathcal{T}}(G[S])$ *of the graph* $G[S]$, $S = \{v_1, v_2, \ldots, v_{n-1}\}$, *on at most* $n-1$ *vertices satisfying Property* 1PC-on-$G[S]$. *Let* $\lambda_{\mathcal{T}}(G[S])$ *be the size of* $\mathcal{P}_{\mathcal{T}}(G[S])$ *and let* v_n *be the terminal vertex*; *then Algorithm* 1PC_INTERVAL *computes a minimum* 1PC $\mathcal{P}_{\mathcal{T}}(G)$ *of the graph G satisfying Property* 1PC-on-G.

Proof Clearly, the size $\lambda'_{\mathcal{T}}(G)$ of a minimum 1PC of G is equal to $\lambda_{\mathcal{T}}(G[S])$ or $\lambda_{\mathcal{T}}(G[S]) + 1$. We distinguish the following cases:

Case 1 When the algorithm processes vertex v_n, it connects v_n to a path, that is, $\lambda_{\mathcal{T}}(G) = \lambda_{\mathcal{T}}(G[S])$. Since v_n is the terminal vertex, the 1PC $\mathcal{P}_{\mathcal{T}}(G)$ is a minimum 1PC of G, that is, $\lambda'_{\mathcal{T}}(G) = \lambda_{\mathcal{T}}(G) = \lambda_{\mathcal{T}}(G[S])$.

We show that the algorithm computes a minimum 1PC $\mathcal{P}_{\mathcal{T}}(G)$ of the graph G having $\varepsilon_i^{(n)}$ endpoints v_κ belonging to different paths with index $\kappa \in (i, n]$, $1 \le i \le n$, such that there is no other minimum 1PC $\mathcal{P}'_{\mathcal{T}}(G)$ having $\varepsilon_i^{\prime(n)}$ endpoints $v_{\kappa'}$ belonging to different paths with index $\kappa' \in (i, n]$ such that $\varepsilon_i^{\prime(n)} > \varepsilon_i^{(n)}$, $1 \le i \le n$.

Suppose that $v_n \in P = (v_n, v_a, \ldots) \in \mathcal{P}_{\mathcal{T}}(G)$. Then, there cannot exist a minimum 1PC $\mathcal{P}'_{\mathcal{T}}(G)$ such that $1 = \varepsilon_{\varrho-1}^{(n)} < \varepsilon_{\varrho-1}^{\prime(n)}$. Assume that there exists a minimum 1PC $\mathcal{P}'_{\mathcal{T}}(G)$ having an index, say, $k-1$, for which we have $\varepsilon_{k-1}^{\prime(n)}$ endpoints $v_{\kappa'}$ belonging to different paths with index $\kappa' \in (k-1, n]$, where $\varepsilon_{k-1}^{\prime(n)} > \varepsilon_{k-1}^{(n)}$. Similarly, to Case 1 of Lemma 4.1, there exists a vertex v_j, $1 \le j < k-1$, such that $\varepsilon_j^{(n)} = \varepsilon_j^{\prime(n)}$.

Suppose that $v_n \in P' = (v_n, v_{a'}, \ldots) \in \mathcal{P}'_{\mathcal{T}}(G)$. If $v_{a'} < v_{j+1}$, then $v_{j+1}v_n \in E(G)$, and, thus, $v_a < v_{j+1}$. Then, if we remove v_n from $\mathcal{P}_{\mathcal{T}}(G)$ and $\mathcal{P}'_{\mathcal{T}}(G)$, we obtain $\varepsilon_{j+1}^{(n-1)} = \varepsilon_{j+1}^{(n)} - 1 = \varepsilon_{j+1}^{\prime(n)} - 2$ and $\varepsilon_{j+1}^{\prime(n-1)} = \varepsilon_{j+1}^{\prime(n)} - 1$; thus, $\varepsilon_{j+1}^{(n-1)} = \varepsilon_{j+1}^{\prime(n-1)} - 1$, a contradiction. On the other hand, if $v_{a'} > v_{j+1}$, then, if we remove v_n from $\mathcal{P}_{\mathcal{T}}(G)$ and $\mathcal{P}'_{\mathcal{T}}(G)$, we obtain $\varepsilon_{j+1}^{(n-1)} = \varepsilon_{j+1}^{(n)} - 1 = \varepsilon_{j+1}^{\prime(n)} - 2$ or $\varepsilon_{j+1}^{(n-1)} = \varepsilon_{j+1}^{(n)} = \varepsilon_{j+1}^{\prime(n)} - 1$. Also, $\varepsilon_{j+1}^{\prime(n-1)} = \varepsilon_{j+1}^{\prime(n)}$; thus, $\varepsilon_{j+1}^{(n-1)} = \varepsilon_{j+1}^{\prime(n-1)} - 1$, a contradiction.

Case 2 When the algorithm processes vertex v_n, it constructs a new trivial path, that is, $\lambda_{\mathcal{T}}(G) = \lambda_{\mathcal{T}}(G[S]) + 1$. Suppose that there exists a 1PC $\mathcal{P}'_{\mathcal{T}}(G)$ of size $\lambda_{\mathcal{T}}(G[S])$. Clearly, vertex v_n cannot belong to a trivial path in $\mathcal{P}'_{\mathcal{T}}(G)$, since removing it results to a 1PC of $G[S]$ of size $\lambda_{\mathcal{T}}(G[S]) - 1$, a contradiction. Thus, let $P = (v_n, v_r, \ldots) \in \mathcal{P}'_{\mathcal{T}}(G)$ be the path containing v_n. Removing vertex v_n from $\mathcal{P}'_{\mathcal{T}}(G)$ results to a 1PC of $G[S]$ of size $\lambda_{\mathcal{T}}(G[S])$, and, thus, minimum, having a neighbor of v_n, that is, vertex v_r, as an endpoint of a path. Since $G[S]$ does not contain the terminal vertex this is a contradiction. Consequently, the 1PC $\mathcal{P}_{\mathcal{T}}(G)$ is a minimum 1PC of G, that is, $\lambda'_{\mathcal{T}}(G) = \lambda_{\mathcal{T}}(G) = \lambda_{\mathcal{T}}(G[S]) + 1$.

We show that the algorithm computes a minimum 1PC $\mathcal{P}_{\mathcal{T}}(G)$ of the graph G having $\varepsilon_i^{(n)}$ endpoints v_κ belonging to different paths with index $\kappa \in (i, n]$, $1 \le i \le n$, such that there is no other minimum 1PC $\mathcal{P}'_{\mathcal{T}}(G)$ having $\varepsilon_i^{\prime(n)}$ endpoints $v_{\kappa'}$ belonging to different paths with index $\kappa' \in (i,n]$ such that $\varepsilon_i^{\prime(n)} > \varepsilon_i^{(n)}$, $1 \le i \le n$.

Since $P = (v_n)$ there cannot exist a minimum 1PC $\mathcal{P}'_{\mathcal{T}}(G)$ such that $1 = \varepsilon_{\varrho-1}^{(n)} < \varepsilon_{\varrho-1}^{\prime(n)}$. Assume that there exists a minimum 1PC $\mathcal{P}'_{\mathcal{T}}(G)$ having an index, say, $k-1$, for which we have $\varepsilon_{k-1}^{\prime(n)}$ endpoints $v_{\kappa'}$ belonging to different paths with index $\kappa' \in (k-1,n]$, where $\varepsilon_{k-1}^{\prime(n)} > \varepsilon_{k-1}^{(n)}$. Suppose that $\varepsilon_{k-1}^{(n)} = x$ and $\varepsilon_{k-1}^{\prime(n)} = x+1$. Similarly, to Case 1 of Lemma 4.1, there exists a vertex v_j, $1 \le j < k-1$, such that $\varepsilon_j^{(n)} = \varepsilon_j^{\prime(n)}$.

Suppose that $v_n \in P' = (v_n, v_{a'}, \ldots) \in \mathcal{P}'_{\mathcal{T}}(G)$; the case where $P' = (v_n)$ is trivial. If $v_{a'} < v_{j+1}$, then $v_{j+1}v_n \in E(G)$, and, thus, vertex v_n would be connected, a contradiction. If $v_{a'} > v_{j+1}$, then, if we remove v_n from $\mathcal{P}_{\mathcal{T}}(G)$ and $\mathcal{P}'_{\mathcal{T}}(G)$, we obtain $\varepsilon_{j+1}^{(n-1)} = \varepsilon_{j+1}^{(n)} - 1 = \varepsilon_{j+1}^{\prime(n)} - 2$ and $\varepsilon_{j+1}^{\prime(n-1)} = \varepsilon_{j+1}^{\prime(n)}$, a contradiction. □

Consequently, we prove the following theorem.

Theorem 4.1 *Let G be an interval graph on n vertices and m edges and let $\mathcal{T} = \{v\}$, where $v \in V(G)$. Algorithm* 1PC_Interval *computes a minimum* 1PC $\mathcal{P}_{\mathcal{T}}(G)$ *of the graph G satisfying Property* 1PC-on-G.

Proof We use induction on n. The basis $n = 1$ is trivial. Assume now that Algorithm 1PC_INTERVAL computes a minimum 1PC $\mathcal{P}_{\mathcal{T}}(G[S])$ of every interval graph $G[S]$, $S = \{v_1, v_2, \ldots, v_{n-1}\}$, on at most $n-1$ vertices having $\varepsilon_i^{(n-1)}$ endpoints v_κ belonging to different paths with index $\kappa \in (i, n-1]$, $1 \le i \le n-1$, such that there is no other minimum 1PC $\mathcal{P}'_{\mathcal{T}}(G[S])$ having $\varepsilon_i^{\prime(n-1)}$ endpoints $v_{\kappa'}$ belonging to different paths with index $\kappa' \in (i, n-1]$ such that $\varepsilon_i^{\prime(n-1)} > \varepsilon_i^{(n-1)}$, $1 \le i < d$, where d is the index of the rightmost endpoint of a path in $\mathcal{P}_{\mathcal{T}}(G[S])$, and, moreover, exactly one of the following holds:

(i) $\varepsilon_i^{\prime(n-1)} \le \varepsilon_i^{(n-1)}$, $d \le i \le n-1$;
(ii) $\varepsilon_i^{\prime(n-1)} = \varepsilon_i^{(n-1)} + 1$, $d \le i < d'$ and $\varepsilon_i^{\prime(n-1)} = \varepsilon_i^{(n-1)}$, $d' \le i \le n-1$, where d' is the index of the rightmost endpoint of a path in $\mathcal{P}'_{\mathcal{T}}(G[S])$, and there exists a vertex v_q such that $\varepsilon_q^{(n-1)} > \varepsilon_q^{\prime(n-1)}$, and there exists no vertex $v_{q'}$ such that $\varepsilon_{q'}^{\prime(n-1)} > \varepsilon_{q'}^{(n-1)}$, $1 \le q, q' < d$.

Let $\lambda_{\mathcal{T}}(G[S])$ be the size of $\mathcal{P}_{\mathcal{T}}(G[S])$. Using Lemmas 4.1–4.6, we show that the algorithm computes a minimum 1PC $\mathcal{P}_{\mathcal{T}}(G)$ of the graph G having $\varepsilon_i^{(n)}$ endpoints v_κ belonging to different paths with index $\kappa \in (i,n]$, $1 \le i \le n$, such that there is no other minimum 1PC $\mathcal{P}'_{\mathcal{T}}(G)$ having $\varepsilon_i^{\prime(n)}$ endpoints $v_{\kappa'}$ belonging to different paths with index $\kappa' \in (i,n]$ such that $\varepsilon_i^{\prime(n)} > \varepsilon_i^{(n)}$, $1 \le i < \varrho$, where ϱ is the index of the rightmost endpoint of a path in $\mathcal{P}_{\mathcal{T}}(G)$, and, moreover, exactly one of the following holds:

(i) $\varepsilon_i'^{(n)} \le \varepsilon_i^{(n)}$, $\varrho \le i \le n$;

(ii) $\varepsilon_i'^{(n)} = \varepsilon_i^{(n)} + 1$, $\varrho \le i < \varrho'$ and $\varepsilon_i'^{(n)} = \varepsilon_i^{(n)}$, $\varrho' \le i \le n$, where ϱ' is the index of the rightmost endpoint of a path in $\mathcal{P}'_{\mathcal{T}}(G)$, and there exists a vertex v_z such that $\varepsilon_z^{(n)} > \varepsilon_z'^{(n)}$, and there exists no vertex $v_{z'}$ such that $\varepsilon_{z'}'^{(n)} > \varepsilon_{z'}^{(n)}$, $1 \le z, z' < \varrho$.

Thus the theorem follows. □

Let us now compute the time and space complexity of Algorithm 1PC_INTERVAL. As mentioned in the description of the algorithm, it constructs the ordering $\pi = (v_1, v_2, \dots, v_n)$ of the vertices of the input graph G and, using a greedy approach on π, computes a minimum 1PC $\mathcal{P}(G)$ of the interval graph G. In particular, it visits the vertices in the ordering $\pi = (v_1, \dots, v_i, \dots, v_n)$ from left to right, and computes a minimum 1PC of the graph $G([S \cup \{v_i\}])$ from a minimum 1PC of $G([S])$, $1 \le i \le n$, where $S = \{v_1, v_2, \dots, v_{i-1}\}$, using the operations `connect`, `insert`, `bridge`, `new_path`, and `connect_break`. It is easy to see that each one of these operations takes time linear to the size of the graph $G[\{v_1, v_2, \dots, v_{i-1}\}]$. Note that, if no vertex is specified to be terminal the algorithm uses only the operations `connect`, `bridge` or `new_path` and, thus, it constructs a minimum PC $\mathcal{P}(G)$ of G in $O(n^2)$ time.

Let G be an interval graph on n vertices and m edges and let $\mathcal{T}$ be a terminal set containing a vertex $v \in V(G)$. Suppose that Algorithm 1PC_INTERVAL processes vertex v_i, $1 \le i \le n$, and t is the index of the terminal vertex; we consider the following cases:

Case $i < t$. The algorithm processes vertex v_i, where $i < t$. In this case, one of the following operations is performed: `connect`, `bridge` or `new_path`. Each operation includes searching for endpoints of paths in $\mathcal{P}_{\mathcal{T}}(G[\{v_1, v_2, \dots, v_{i-1}\}])$ that are neighbors of v_i. Thus, a minimum 1PC of $G[\{v_1, v_2, \dots, v_i\}]$ is constructed in time linear to the size of $G[\{v_1, v_2, \dots, v_{i-1}\}]$.

Case $i = t$. The algorithm processes vertex v_t. Then, either operation `connect` or operation `new_path` is performed; specifically, the procedure `process_terminal` is searching for the leftmost endpoint, say, v_ℓ, of a path in $\mathcal{P}_{\mathcal{T}}(G[\{v_1, v_2, \dots, v_{t-1}\}])$ being a neighbor of v_t, and, either connects v_t to v_ℓ or creates a new trivial path. Thus, a minimum 1PC of $G[\{v_1, v_2, \dots, v_t\}]$ is constructed in time linear to the size of $G[\{v_1, v_2, \dots, v_{t-1}\}]$.

Case $i > t$. The algorithm processes vertex v_i, where $t < i$. Unless the computation of a minimum PC of $G[\{v_1, v_2, \dots, v_{i-1}\} - \{v_t\}]$ is required or a minimum 1PC of $G[\{v_1, \dots, v_t, \dots, v_{i-1}\}]$ is reconstructed, the algorithm processes vertex v_i using the operations `connect`, `insert`, `bridge`, `new_path`, and `connect_break`. Since each one of these operations takes $O(i)$ time, in this case a minimum 1PC of $G[\{v_1, \dots, v_t, \dots, v_i\}]$ is computed in $O(i)$ time. In the case where a minimum PC of $G[\{v_1, v_2, \dots, v_{i-1}\} - \{v_t\}]$ is computed or a minimum 1PC of $G[\{v_1, \dots, v_t, \dots, v_{i-1}\}]$ is reconstructed, the algorithm computes a minimum 1PC of $G[\{v_1, v_2, \dots, v_i\}]$ in $O(i^2)$ time. Thus, a minimum 1PC of $G[\{v_1, v_2, \dots, v_i\}]$ is constructed in time quadratic to the size of $G[\{v_1, v_2, \dots, v_{i-1}\}]$.

Taking into consideration the time complexity of the operations performed by the algorithm in each case, we conclude that it computes a minimum 1PC $\mathcal{P}_{\mathcal{T}}(G)$ of G

in $O(n^3)$ time and, since no additional space is needed, requires linear space. Recall that the ordering π of the vertices is constructed in linear time [24]. Hence, we can state the following result.

Theorem 4.2 *Let G be an interval graph on n vertices and let $\mathcal{T}$ be a subset of $V(G)$ containing a single vertex. A minimum 1-fixed-endpoint path cover of G with respect to $\mathcal{T}$ can be computed in $O(n^3)$ time.*

5 Related Results on Convex and Biconvex Graphs

Based on the results for the 1PC problem on interval graphs, and also on the reduction described by Müller in [20], we study the HP and 1HP problems on convex and biconvex graphs. A bipartite graph $G(X, Y; E)$ is called *X-convex* (or simply convex) if there exists an ordering of its vertices such that for all $y \in Y$ the vertices of $N(y)$ are consecutive [20]; G is *biconvex* if it is convex on both X and Y.

In this section, we solve the HP and 1HP problems on a biconvex graph $G(X, Y; E)$. Moreover, we show that the HP problem on an X-convex graph $G(X, Y; E)$ on n vertices can be solved in $O(n^4)$ time if $|X| = |Y|$ or $|X| - |Y| = 1$ and a 1HP starting at vertex u, if there exists, can be computed in $O(n^3)$ time if ($|X| = |Y|$ and $u \in Y$) or $|X| - |Y| = 1$.

We next describe an algorithm for the HP problem on a biconvex graph $G(X, Y; E)$. To simplify our description we use two functions, namely `Minimum_HP(`G`)` and `Minimum_1PC(`G, u`)`: the function `Minimum_HP(`G`)` corresponds to the algorithm for computing a minimum path cover of an interval graph G described in [2], while `Minimum_1PC(`G, u`)` corresponds to the algorithm for computing a minimum path cover with specified endpoint u of an interval graph G described in this paper.

Algorithm HP_BICONVEX
Input: a biconvex graph $G(X, Y; E)$ on n vertices;
Output: a Hamiltonian path of G, if one exists:

1. `if` $||X| - |Y|| > 1$ `then return` G does not have a Hamiltonian path;
2. `if` $|X| = |Y|$ `then`
   `construct` the interval graph G' such that:
      $V(G') = X \cup Y$, $E(G') = E \cup E_Y$, where E_Y is as follows:
      $y_1 y_2 \in E_Y$ iff $y_1, y_2 \in Y$ and $N(y_1) \cap N(y_2) \neq \emptyset$;
   `if` there exists $y_j \in Y$ such that $|N(y_j)| = 1$ `then`
      $\mathcal{P}_{\mathcal{T}}(G) \leftarrow$ `Minimum_1PC(`G', y_j`);`
      `if` $\lambda_{\mathcal{T}}(G) = 1$ `then return` $\mathcal{P}_{\mathcal{T}}(G)$;
      `else return` G does not have a Hamiltonian path;
   `else`
      `for` $i = 1$ `to` $|Y|$ `do`
         $\mathcal{P}_{\mathcal{T}}(G) \leftarrow$ `Minimum_1PC(`G', y_i`);`
         `if` $\lambda_{\mathcal{T}}(G) = 1$ `then return` $\mathcal{P}_{\mathcal{T}}(G)$;
      `end-for;`
      `return` G does not have a Hamiltonian path;

3. `if` $|X|-|Y|=1$ `then`
   $\mathcal{P}_{\mathcal{T}}(G) \leftarrow$ `Minimum_HP(`G'`);`
   `if` $\lambda_{\mathcal{T}}(G)=1$ `then return` $\mathcal{P}_{\mathcal{T}}(G)$;
   `else return` G does not have a Hamiltonian path;
4. `if` $|Y|-|X|=1$ `then`
   `construct` the interval graph G' such that:
      $V(G')=X\cup Y$, $E(G')=E\cup E_X$, where E_X is as follows:
      $x_1x_2\in E_X$ iff $x_1,x_2\in X$ and $N(x_1)\cap N(x_2)\neq\emptyset$;
   $\mathcal{P}_{\mathcal{T}}(G) \leftarrow$ `Minimum_HP(`G'`);`
   `if` $\lambda_{\mathcal{T}}(G)=1$ `then return` $\mathcal{P}_{\mathcal{T}}(G)$;
   `else return` G does not have a Hamiltonian path;

End_of_Algorithm HP_BICONVEX.

Observation 5.1 Uehara and Uno in [31] claim that the HP problem on a biconvex graph $G(X,Y;E)$ on n vertices can be solved in $O(n^2)$ time even if $|X|=|Y|$. Specifically, they claim that G has an HP iff the interval graph G' has an HP, where G' is an interval graph such that $V(G')=X\cup Y$ and $E(G')=E\cup E_Y$, where E_Y is as follows: $y_1y_2\in E_Y$ iff $y_1,y_2\in Y$ and $N(y_1)\cap N(y_2)\neq\emptyset$. However, this is not true, since there exists a counterexample, which is presented in Fig. 7. Indeed, the biconvex graph G of Fig. 7 does not have an HP while for G' we have $P=(x_1,y_2,x_2,y_1,y_4,x_3,y_3,x_4)$. Suppose that we construct an interval graph G' such that $V(G')=X\cup Y$ and $E(G')=E\cup E_X$, where E_X is as follows: $x_1x_2\in E_X$ iff $x_1,x_2\in X$ and $N(x_1)\cap N(x_2)\neq\emptyset$. Then, G' has an HP, that is, $P=(y_4,x_3,y_3,x_4,x_1,y_2,x_2,y_1)$. Thus, there exists no algorithm with time complexity $O(n^2)$ and we can state the following result.

Theorem 5.1 *The Hamiltonian path problem on a biconvex graph G on n vertices can be solved in $O(n^4)$ time.*

Similarly, we show that the Hamiltonian path problem on an X-convex graph $G(X,Y;E)$ on n vertices can be solved in $O(n^4)$ time when $|X|=|Y|$ or $|X|-|Y|=1$. It is easy to see that if $|Y|-|X|=1$ then the X-convex graph $G(X,Y;E)$ has a Hamiltonian path iff the interval graph G' has a 2HP between any two vertices of Y. Thus, we can state the following result.

Corollary 5.1 *The Hamiltonian path problem on an X-convex graph $G(X,Y;E)$ on n vertices can be solved in $O(n^4)$ time if $|X|=|Y|$ or $|X|-|Y|=1$.*

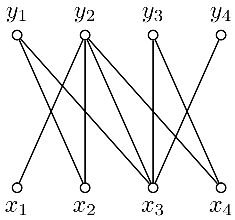


Fig. 7 A biconvex graph $G=(X,Y;E)$ with $|X|=|Y|$

We next describe an algorithm for the 1HP problem on a biconvex graph $G(X, Y; E)$. Recall that the function `Minimum_1PC(`G`,`u`)` corresponds to the algorithm for computing a minimum path cover with specified endpoint u of an interval graph G described in this paper.

Algorithm 1HP_BICONVEX
Input: a biconvex graph $G(X, Y; E)$ on n vertices and a vertex $y_t \in Y$;
Output: a Hamiltonian path of G starting at vertex y_t, if one exists:

1. `if` $||X| - |Y|| > 1$ `then return` G does not have a 1HP;
2. `if` $|X| = |Y|$ `then`
   `construct` the interval graph G' such that:
     $V(G') = X \cup Y$, $E(G') = E \cup E_Y$, where E_Y is as follows:
     $y_1 y_2 \in E_Y$ iff $y_1, y_2 \in Y$ and $N(y_1) \cap N(y_2) \neq \emptyset$;
   $\mathcal{P}_{\mathcal{T}}(G) \leftarrow$ `Minimum_1PC(`G', y_t`);`
   `if` $\lambda_{\mathcal{T}}(G) = 1$ `then return` $\mathcal{P}_{\mathcal{T}}(G)$;
   `else return` G does not have a 1HP;
3. `if` $|X| - |Y| = 1$ `then return` G does not have a 1HP;
4. `if` $|Y| - |X| = 1$ `then`
   `construct` the interval graph G' such that:
     $V(G') = X \cup Y$, $E(G') = E \cup E_X$, where E_X is as follows:
     $x_1 x_2 \in E_X$ iff $x_1, x_2 \in X$ and $N(x_1) \cap N(x_2) \neq \emptyset$;
   $\mathcal{P}_{\mathcal{T}}(G) \leftarrow$ `Minimum_1PC(`G', y_t`);`
   `if` $\lambda_{\mathcal{T}}(G) = 1$ `then return` $\mathcal{P}_{\mathcal{T}}(G)$;
   `else return` G does not have a 1HP;

End_of_Algorithm 1HP_BICONVEX.

Since the function `Minimum_1PC` requires $O(n^3)$ time to compute a 1HP of an interval graph on n vertices and the graph G' can be constructed in $O(|X \cup Y|^2)$ time [20], Algorithm 1HP_BICONVEX returns a 1HP, if there exists, of a biconvex graph on n vertices in $O(n^3)$ time. Hence, we can state the following result.

Theorem 5.2 *Let G be a biconvex graph on n vertices and let u be a vertex of $V(G)$. The* 1HP *problem on G and u can be solved in $O(n^3)$ time.*

Let $G(X, Y; E)$ be an X-convex graph on n vertices and let u be a vertex of $V(G)$. Similarly, we show that the 1HP problem on G and u can be solved in $O(n^3)$ time when ($|X| = |Y|$ and $u \in Y$) or $|X| - |Y| = 1$. Clearly, if $|Y| - |X| = 1$ and $u \in X$ then G does not have a 1HP. It is easy to see that if ($|Y| - |X| = 1$ and $u \in Y$) or ($|X| = |Y|$ and $u \in X$) then the X-convex graph $G(X, Y; E)$ has a Hamiltonian path if the interval graph G' has a 2HP between u and a vertex of Y. Thus, we can state the following result.

Corollary 5.2 *Let $G(X, Y; E)$ be an X-convex graph on n vertices and let u be a vertex of $V(G)$. The* 1HP *problem on G and u can be solved in $O(n^3)$ time if ($|X| = |Y|$ and $u \in Y$) or $|X| - |Y| = 1$. If $|Y| - |X| = 1$ and $u \in X$ then G does not have a* 1HP.

6 Concluding Remarks

In this paper we presented a polynomial-time algorithm for the 1PC problem on interval graphs and, thus, we answered the question left open by Damaschke [8] of whether the 1HP problem is polynomially solvable on interval graphs. We also presented polynomial-time algorithms for the HP and 1HP problems on biconvex graphs, and showed that the HP problem on X-convex graphs $G(X, Y, E)$ with $|Y| - |X| = 1$, which was left open in [31], has polynomial solution iff the 2HP problem on interval graphs is polynomial.

An interesting open question is whether the k-fixed-endpoint path cover problem (kPC) can be polynomially solved on interval graphs. Note that, the kPC problem generalizes both the 1HP and 2HP problems; the complexity status of the 2HP problem on interval graphs remains an open question.

References

1. Adhar, G.S., Peng, S.: Parallel algorithm for path covering, Hamiltonian path, and Hamiltonian cycle in cographs. In: International Conference on Parallel Processing, vol. III: Algorithms and Architecture, pp. 364–365. Pennsylvania State University Press, Pennsylvania (1990)
2. Arikati, S.R., Rangan, C.P.: Linear algorithm for optimal path cover problem on interval graphs. Inf. Process. Lett. **35**, 149–153 (1990)
3. Asdre, K., Nikolopoulos, S.D.: A linear-time algorithm for the k-fixed-endpoint path cover problem on cographs. Networks **50**, 231–240 (2007)
4. Asdre, K., Nikolopoulos, S.D.: A polynomial solution for the k-fixed-endpoint path cover problem on proper interval graphs. In: Proc. of the 18th International Conference on Combinatorial Algorithms (IWOCA'07), Lake Macquarie, Newcastle, Australia (2007)
5. Bertossi, A.A.: Finding Hamiltonian circuits in proper interval graphs. Inf. Process. Lett. **17**, 97–101 (1983)
6. Bertossi, A.A., Bonuccelli, M.A.: Finding Hamiltonian circuits in interval graph generalizations. Inf. Process. Lett. **23**, 195–200 (1986)
7. Brandstädt, A., Le, V.B., Spinrad, J.: Graph Classes—A Survey. SIAM Monographs in Discrete Mathematics and Applications. SIAM, Philadelphia (1999)
8. Damaschke, P.: Paths in interval graphs and circular arc graphs. Discrete Math. **112**, 49–64 (1993)
9. Fortune, S., Hopcroft, J.E., Wyllie, J.: The directed subgraph homeomorphism problem. J. Theor. Comput. Sci. **10**, 111–121 (1980)
10. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. Freeman, San Francisco (1979)
11. Garey, M.R., Johnson, D.S., Tarjan, R.E.: The planar Hamiltonian circuit problem is NP-complete. SIAM J. Comput. **5**, 704–714 (1976)
12. Golumbic, M.C.: Algorithmic Graph Theory and Perfect Graphs. Academic Press, New York (1980) (2nd edn., in Ann. Discrete Math. **57**, Elsevier, 2004)
13. Hsieh, S.Y.: An efficient parallel strategy for the two-fixed-endpoint Hamiltonian path problem on distance-hereditary graphs. J. Parallel Distributed Comput. **64**, 662–685 (2004)
14. Hsieh, S.Y., Ho, C.W., Hsu, T.S., Ko, M.T.: The Hamiltonian problem on distance-hereditary graphs. Discrete Appl. Math. **154**, 508–524 (2006)
15. Hung, R.W., Chang, M.S.: Linear-time algorithms for the Hamiltonian problems on distance-hereditary graphs. Theor. Comput. Sci. **341**, 411–440 (2005)
16. Hung, R.W., Chang, M.S.: Solving the path cover problem on circular-arc graphs by using an approximation algorithm. Discrete Appl. Math. **154**, 76–105 (2006)
17. Karp, R.M.: On the complexity of combinatorial problems. Networks **5**, 45–68 (1975)
18. Keil, J.M.: Finding Hamiltonian circuits in interval graphs. Inf. Process. Lett. **20**, 201–206 (1985)
19. Lin, R., Olariu, S., Pruesse, G.: An optimal path cover algorithm for cographs. Comput. Math. Appl. **30**, 75–83 (1995)

20. Müller, H.: Hamiltonian circuits in chordal bipartite graphs. Discrete Math. **156**, 291–298 (1996)
21. Nakano, K., Olariu, S., Zomaya, A.Y.: A time-optimal solution for the path cover problem on cographs. Theor. Comput. Sci. **290**, 1541–1556 (2003)
22. Nikolopoulos, S.D.: Parallel algorithms for Hamiltonian problems on quasi-threshold graphs. J. Parallel Distributed Comput. **64**, 48–67 (2004)
23. Park, J.H.: One-to-many disjoint path covers in a graph with faulty elements. In: Proc. of the 10th International Computing and Combinatorics Conference (COCOON'04), pp. 392–401 (2004)
24. Ramalingam, G., Rangan, C.P.: A unified approach to domination problems on interval graphs. Inf. Process. Lett. **27**, 271–274 (1988)
25. Robertson, N., Seymour, P.D.: Graph minors. XIII. The disjoint paths problem. J. Comb. Theory Ser. B **63**, 65–110 (1995)
26. Seymour, P.D.: Disjoint paths in graphs. Discrete Math. **29**, 293–309 (1980)
27. Shiloach, Y.: A polynomial solution to the undirected two paths problem. J. Assoc. Comput. Mach. **27**, 445–456 (1980)
28. Srikant, R., Sundaram, R., Singh, K.S., Rangan, C.P.: Optimal path cover problem on block graphs and bipartite permutation graphs. Theor. Comput. Sci. **115**, 351–357 (1993)
29. Suzuki, Y., Kaneko, K., Nakamori, M.: Node-disjoint paths algorithm in a transposition graph. IEICE Trans. Inf. Syst. **E89-D**, 2600–2605 (2006)
30. Thomassen, C.: 2-linked graphs. Eur. J. Comb. **1**, 371–378 (1980)
31. Uehara, R., Uno, Y.: On computing longest paths in small graph classes. Int. J. Found. Comput. Sci. **18**, 911–930 (2007)