A Linear-Time Algorithm for the k-Fixed-Endpoint Path Cover Problem on Cographs

Katerina Asdre and Stavros D. Nikolopoulos
Department of Computer Science, University of Ioannina, GR-45110 Ioannina, Greece

In this paper, we study a variant of the path cover problem, namely, the k-fixed-endpoint path cover problem. 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 k-fixed-endpoint path cover problem is to find a k-fixed-endpoint path cover of G of minimum cardinality; note that, if $\mathcal{T}$ is empty, that is, $k = 0$, the stated problem coincides with the classical path cover problem. We show that the k-fixed-endpoint path cover problem can be solved in linear time on the class of cographs. More precisely, we first establish a lower bound on the size of a minimum k-fixed-endpoint path cover of a cograph and prove structural properties for the paths of such a path cover. Then, based on these properties, we describe an algorithm which, for a cograph G on n vertices and m edges, computes a minimum k-fixed-endpoint path cover of G in linear time, that is, in $O(n + m)$ time. The proposed algorithm is simple, requires linear space, and also enables us to solve some path cover related problems, such as the 1HP and 2HP, on cographs within the same time and space complexity. © 2007 Wiley Periodicals, Inc. NETWORKS, Vol. 50(4), 231–240 2007

Keywords: perfect graphs; complement reducible graphs; cographs; cotree; path cover; fixed-endpoint path cover; linear-time algorithms

1. INTRODUCTION

1.1. 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 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, 18, 22]); it is well known that the path cover problem and many of its variants are NP-complete in general graphs [8]. 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.

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 [8].

The path cover problem and several variants of it have numerous algorithmic applications in many fields. Some that have received both theoretical and practical attention are in the content of communication and/or transposition networks [23]. In such problems, we are given a graph (network) G and

(Problem A) a set $\mathcal{T}$ of $k = 2\lambda$ vertices of G, and the objective is to determine whether G admits a path cover of size λ that contains paths connecting pairs of vertices of $\mathcal{T}$, that is, G admits λ vertex-disjoint paths with both their endpoints in $\mathcal{T}$ (note that, the endpoints of a path P are the first vertex and the last vertex visited by P), or

(Problem B) a set $\mathcal{T}$ of $\lambda = k/2$ pairs of vertices of G (source-sink pairs), and the objective is to determine whether G admits for each pair (a_i, b_i), $1 \leq i \leq \lambda$, a path connecting a_i to b_i such that the set of λ paths forms a path cover.

In the case where $k = 2$, both problems A and B coincide with the 2HP. In [7], 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; until now the complexities of 1HP and 2HP keep their difficulties even in the small subclass of split interval graphs—no polynomial algorithm is known.

Motivated by the above issues we state a variant of the path cover problem, namely, the k-fixed-endpoint path cover

Received January 2006; accepted February 2007
Correspondence to: S. D. Nikolopoulos; e-mail: stavros@cs.uoi.gr
DOI 10.1002/net.20200
Published online in Wiley InterScience (www.interscience.wiley.com).

problem (kPC), which generalizes both 1HP and 2HP, and also problem A.
(Problem kPC) Let G be a graph and let $\mathcal{T}$ be a set of k vertices of $V(G)$. A *k-fixed-endpoint path cover* of the graph G with respect to $\mathcal{T}$ is a path cover of G such that all vertices in $\mathcal{T}$ are endpoints of paths in the path cover; a *minimum k-fixed-endpoint path cover* of G with respect to $\mathcal{T}$ is a k-fixed-endpoint path cover of G with minimum cardinality; the *k-fixed-endpoint path cover problem* (kPC) is to find a minimum k-fixed-endpoint path cover of the graph G.

We show that the k-fixed-endpoint path cover problem (kPC) has a polynomial-time solution in the class of complement reducible graphs, or cographs [6, 17]. The class of cographs is defined recursively as follows: (i) a single vertex graph is a cograph; (ii) if G is a cograph then its complement $\overline{G}$ is also a cograph; (iii) if G_1 and G_2 are cographs satisfying $V(G_1) \cap V(G_2) = \emptyset$, then their union $G_1 \cup G_2$ is also a cograph. Thus, cographs are formed from a single vertex under the closure of the operations of union and complement. Cographs were independently discovered under various names and were shown to have the following two remarkable properties: they are P_4 restricted graphs and they have a unique tree representation (see [17]). This tree, called the *co-tree*, forms the basis for fast algorithms for problems such as isomorphism, coloring, clique detection, clusters, minimum weight dominating sets [4, 5], and also for the path cover problem [18, 20].

1.2. Contribution

In this paper, we study the complexity status of the k-fixed-endpoint path cover problem (kPC) on the class of cographs, and show that this problem can be solved in polynomial time when the input is a cograph. More precisely, we establish a lower bound on the size of a minimum k-fixed-endpoint path cover of a cograph G on n vertices and m edges. We then define path operations, and prove structural properties for the paths of such a path cover, which enable us to describe a simple algorithm for the kPC problem. The proposed algorithm runs in time linear in the size of the input graph G, that is, in $O(n + m)$ time, and requires linear space. To the best of our knowledge, this is the first linear-time algorithm for solving the kPC problem on the class of cographs.

The proposed algorithm for the kPC problem can also be used to solve the 1HP and 2HP problems on cographs within the same time and space complexity. Moreover, we have designed our algorithm so that it produces a minimum k-fixed-endpoint path cover of a cograph G that contains a large number of paths with both their endpoints in $\mathcal{T}$ (we can easily find a graph G and a set $\mathcal{T}$ of k vertices of $V(G)$ so that G admits two minimum k-fixed-endpoint path covers with different numbers of paths having both their endpoints in $\mathcal{T}$; for example, consider the graph G with vertex set $V(G) = \{a, b, c, d\}$, edge set $E(G) = \{ab, bc, ac, cd\}$, and $\mathcal{T} = \{a, b\}$). Thus, we can also use our algorithm to solve problem A on cographs within the same time and space complexity.

1.3. Related Work

The class of cographs has been extensively studied and several sequential and/or parallel algorithms for recognition and for classical combinatorial optimization problems have been proposed. Corneil et al. [6] proposed a linear-time recognition algorithm for cographs. Jung [16] studied the existence of a Hamiltonian path or cycle in a cograph, while Lin et al. [18] proposed an optimal algorithm for the path cover problem on cographs. Nakano et al. [20] 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. Furthermore, quite recently Nikolopoulos [21] solved the Hamiltonian problem on quasi-threshold graphs (a subclass of cographs) in $O(\log n)$ time using $O(n+m)$ processors on a PRAM model. Sequential algorithms for optimization problems on other related classes of graphs (superclasses of cographs) have also been proposed: Giakoumakis et al. [9] solved the recognition problem and also the problems of finding the clique number, the stability number, and the chromatic number for P_4-sparse graphs [10] (a proper superclass of cographs) in linear sequential time. Hochstättler and Tinhofer [11] presented a sequential algorithm for the path cover problem on this class of graphs, which runs in $f(n) + O(n)$ time, where $f(n)$ is the time complexity for the construction of a tree representation of a P_4-sparse graph. Also, Giakoumakis et al. [9] studied hamiltonicity properties for the class of P_4-*tidy* graphs (a proper superclass of P_4-sparse graphs); see also [3]. Recently, Hsieh et al. [13] 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 [14], while Hsieh [12] 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 [2, 15, 22].

1.4. Road Map

The paper is organized as follows. In Section 2 we establish the notation and related terminology, and we present background results. In Section 3 we describe our linear-time algorithm for the kPC problem, while in Section 4 we prove its correctness and compute its time and space complexity. Finally, in Section 5 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 $G[S]$.

2.1. The Co-tree

The cographs admit a tree representation unique up to isomorphism. Specifically, we can associate with every cograph G a unique rooted tree $T_{\text{co}}(G)$ called the co-tree (or, modular decomposition tree [19]) of G having the following properties:

1. Every internal node of $T_{\text{co}}(G)$ has at least two children;
2. The internal nodes of $T_{\text{co}}(G)$ are labeled by either P (P-node) or S (S-node) in such a way that the labels alternate along every path in $T_{\text{co}}(G)$ starting at the root;
3. Each leaf of $T_{\text{co}}(G)$ corresponds to a vertex in V such that $(x, y) \in E$ if and only if the lowest common ancestor of the leaves corresponding to x and y is an S-node.

It is shown that for every cograph G the co-tree $T_{\text{co}}(G)$ is unique up to isomorphism and it can be constructed sequentially in linear time [4,6].

For convenience and ease of presentation, we binarize the co-tree $T_{\text{co}}(G)$ in such a way that each of its internal nodes has exactly two children [18,20]. Let t be an internal node of $T_{\text{co}}(G)$ with children $t_1, t_2, \ldots, t_k$ where $k \geq 3$. We replace node t by $k-1$ nodes $t'_1, t'_2, \ldots, t'_{k-1}$ such that t'_1 has children t_1 and t_2 and each t'_i $(2 \leq i < k)$ has children t'_{i-1} and t_{i+1}. We shall refer to the binarized version of $T_{\text{co}}(G)$ as the modified co-tree of G and will denote it by $T(G)$. Thus, the left and right child of an internal node t of $T(G)$ will be denoted by t_ℓ and t_r, respectively.

Let t be an internal node of $T(G)$. Then $G[t]$ is the subgraph of G induced by the subset V_t of the vertex set $V(G)$, which contains all the vertices of G that have as common ancestor in $T(G)$ the node t. For simplicity, we will denote by V_ℓ and V_r the vertex sets $V(G[t_\ell])$ and $V(G[t_r])$, respectively.

2.2. Cographs and the kPC Problem

Let G be a cograph, $\mathcal{T}$ be a set of k vertices of $V(G)$, and let $\mathcal{P}_\mathcal{T}(G)$ be a minimum k-fixed-endpoint path cover of G with respect to $\mathcal{T}$ of size $\lambda_\mathcal{T}$; note that the size of $\mathcal{P}_\mathcal{T}(G)$ is the number of paths it contains. The vertices of the set $\mathcal{T}$ are called *terminal* vertices, and the set $\mathcal{T}$ is called the *terminal set* of G, while those of $V(G) - \mathcal{T}$ are called *non-terminal or free* vertices. Thus, the set $\mathcal{P}_\mathcal{T}(G)$ contains three types of paths, which we call *terminal*, *semi-terminal*, and *nonterminal or free* paths:

(i) a *terminal path P_t* consists of at least two vertices and both its endpoints, say, u and v, are terminal vertices, that is, $u, v \in \mathcal{T}$;
(ii) a *semi-terminal path P_s* is a path having one endpoint in $\mathcal{T}$ and the other in $V(G) - \mathcal{T}$; if P_s consists of only one vertex (trivial path), say, u, then $u \in \mathcal{T}$;
(iii) a *non-terminal or free path P_f* is a path having both its endpoints in $V(G) - \mathcal{T}$; if P_f consists of only one vertex, say, u, then $u \in V(G) - \mathcal{T}$.

Note that all the internal vertices of the paths of $\mathcal{P}_\mathcal{T}(G)$ are free vertices. Moreover, a semi-terminal path may consist of only one vertex which is a terminal vertex, while a terminal path contains at least two vertices. The set of the non-terminal paths in a minimum kPC of the graph G is denoted by N, while S and T denote the sets of the semi-terminal and terminal paths, respectively. Thus, we have

$$\lambda_\mathcal{T} = |N| + |S| + |T| \tag{1}$$

From the definition of the k-fixed-endpoint path cover problem (kPC), we can easily conclude that the number of paths in a minimum kPC cannot be less than the number of terminal vertices divided by two. Furthermore, since each semi-terminal path contains one terminal vertex and each terminal path contains two, the number of terminal vertices is equal to $|S| + 2|T|$. Thus, the following proposition holds:

Proposition 2.1. *Let G be a cograph and let $\mathcal{T}$ be a terminal set of G. Then $|\mathcal{T}| = |S| + 2|T|$ and $\lambda_\mathcal{T} \geq \lceil \frac{|\mathcal{T}|}{2} \rceil$.*

Clearly, the size of a kPC of a cograph G, as well as the size of a minimum kPC of G, is less than or equal to the number of vertices of G, that is, $\lambda_\mathcal{T} \leq |V(G)|$. Let $F(V(G))$ be the set of the free vertices of G; hereafter, $F(V) = F(V(G))$. Then we have the following proposition:

Proposition 2.2. *Let G be a cograph and let $\mathcal{T}$ be a terminal set of G. If $\lambda_\mathcal{T}$ is the size of a minimum kPC of G, then $\lambda_\mathcal{T} \leq |F(V)| + |\mathcal{T}|$.*

Let t be an internal node of the tree $T(G)$. Then $\lambda_\mathcal{T}(t)$ denotes the number of paths in a minimum kPC of the graph $G[t]$, and let t_ℓ and t_r be the left and the right child of node t, respectively. We denote by $\mathcal{T}_\ell$ and $\mathcal{T}_r$ the terminal vertices in V_ℓ and V_r, respectively, where $V_\ell = V(G[t_\ell])$ and $V_r = V(G[t_r])$. Let N_ℓ, S_ℓ, and T_ℓ be the sets of the non-terminal, semi-terminal and terminal paths in a minimum kPC of $G[t_\ell]$, respectively. Similarly, let N_r, S_r, and T_r be the sets of the non-terminal, semi-terminal and terminal paths in a minimum kPC of $G[t_r]$, respectively. Obviously, Equation (1) holds for $G[t]$ as well, with t being either an S-node or a P-node, that is,

$$\lambda_\mathcal{T}(t) = |N_t| + |S_t| + |T_t| \tag{2}$$

where N_t, S_t, and T_t are the sets of the non-terminal, the semi-terminal and the terminal paths in a minimum kPC of $G[t]$, respectively. If t is a P-node, then a minimum kPC $\mathcal{P}_\mathcal{T}(t)$ of $G[t]$ is $\mathcal{P}_\mathcal{T}(t) = \mathcal{P}_\mathcal{T}(t_\ell) \cup \mathcal{P}_\mathcal{T}(t_r)$, where $\mathcal{P}_\mathcal{T}(t_\ell)$ and $\mathcal{P}_\mathcal{T}(t_r)$ are minimum kPCs corresponding to $G[t_\ell]$ and $G[t_r]$, respectively, and $\lambda_\mathcal{T}(t) = \lambda_\mathcal{T}(t_\ell) + \lambda_\mathcal{T}(t_r)$. Furthermore, for the case of a P-node we have

$$|N_t| = |N_\ell| + |N_r|$$
$$|S_t| = |S_\ell| + |S_r|$$
$$|T_t| = |T_\ell| + |T_r|$$

Thus, we focus on computing a minimum kPC of the graph $G[t]$ for the case where t is an S-node.

Before describing our algorithm, we establish a lower bound on the size $\lambda_{\mathcal{T}}(t)$ of a minimum kPC $\mathcal{P}_{\mathcal{T}}(t)$ of a graph $G[t]$. More precisely, we prove the following lemma.

Lemma 2.1. *Let t be an internal node of $T(G)$ and let $\mathcal{P}_{\mathcal{T}}(t_\ell)$ and $\mathcal{P}_{\mathcal{T}}(t_r)$ be a minimum kPC of $G[t_\ell]$ and $G[t_r]$, respectively. Then $\lambda_{\mathcal{T}}(t) \geq \max\{\lceil\frac{|\mathcal{T}_t|}{2}\rceil, \lambda_{\mathcal{T}}(t_\ell) - |F(V_r)|, \lambda_{\mathcal{T}}(t_r) - |F(V_\ell)|\}$.*

Proof. Clearly, according to Proposition 2.1 and since $G[t]$ is a cograph, we have $\lambda_{\mathcal{T}}(t) \geq \lceil\frac{|\mathcal{T}_t|}{2}\rceil$. We will prove that $\lambda_{\mathcal{T}}(t) \geq \lambda_{\mathcal{T}}(t_\ell) - |F(V_r)|$. Assume that $\lambda_{\mathcal{T}}(t) < \lambda_{\mathcal{T}}(t_\ell) - |F(V_r)|$. Consider removing from this path cover all the vertices in V_r. What results is a set of paths which is clearly a kPC for $G[t_\ell]$. Since the removal of a free vertex in $F(V_r)$ will increase the number of paths by at most one, we obtain a kPC of $G[t_\ell]$ of size at most $\lambda_{\mathcal{T}}(t) + |F(V_r)|$. The assumption $\lambda_{\mathcal{T}}(t) < \lambda_{\mathcal{T}}(t_\ell) - |F(V_r)|$ guarantees that $\lambda_{\mathcal{T}}(t) + |F(V_r)| < \lambda_{\mathcal{T}}(t_\ell)$, contradicting the minimality of $\mathcal{P}_{\mathcal{T}}(t_\ell)$. Using similar arguments we can show that $\lambda_{\mathcal{T}}(t) \geq \lambda_{\mathcal{T}}(t_r) - |F(V_\ell)|$. Hence, the lemma follows. ∎

We next define four operations on paths of a minimum kPC of the graphs $G[t_\ell]$ and $G[t_r]$, namely *break*, *connect*, *bridge* and *insert* operations; these operations are illustrated in Figure 1.

- *Break* operation: Let $P = [p_1, p_2, \ldots, p_k]$ be a path of $\mathcal{P}_{\mathcal{T}}(t_r)$ or $\mathcal{P}_{\mathcal{T}}(t_\ell)$ of length k. We say that we *break* the path P in two paths, say, P_1 and P_2, if we delete an arbitrary edge of P, say the edge p_ip_{i+1} $(1 \leq i < k)$, in order to obtain two paths which are $P_1 = [p_1, \ldots, p_i]$ and $P_2 = [p_{i+1}, \ldots, p_k]$. Note that we can break the path P in at most k trivial paths.
- *Connect* operation: Let $P_1 = [p_1, \ldots, p_1']$ be a non-terminal or a semi-terminal path of $\mathcal{P}_{\mathcal{T}}(t_\ell)$ (resp. $\mathcal{P}_{\mathcal{T}}(t_r)$) and let $P_2 = [p_2, \ldots, p_2']$ be a non-terminal or a semi-terminal path of $\mathcal{P}_{\mathcal{T}}(t_r)$ (resp. $\mathcal{P}_{\mathcal{T}}(t_\ell)$). We say that we *connect* the path P_1 with the path P_2, if we add an edge which joins two free endpoints of the two paths.
- *Bridge* operation: Let $P_1 = [p_1, \ldots, p_1']$ and $P_2 = [p_2, \ldots, p_2']$ be two paths of the set $N_\ell \cup S_\ell$ (resp. $N_r \cup S_r$) and let $P_3 = [p_3, \ldots, p_3']$ be a non-terminal path of the set N_r (resp. N_ℓ). We say that we *bridge* the two paths P_1 and P_2 using path P_3 if we connect the free endpoint of P_1 with one endpoint of P_3 and the free endpoint of P_2 with the other endpoint of P_3. The result is a path having both endpoints in $G[t_\ell]$ (resp. $G[t_r]$).
- *Insert* operation: Let $P_1 = [t_1, p_1, \ldots, p_1', t_1']$ be a terminal path of the set T_ℓ (resp. T_r) and let $P_2 = [p_2, \ldots, p_2']$ be a non-terminal path of the set N_r (resp. N_ℓ). We say that we *insert* the path P_2 into P_1, if we replace the first edge of P_1, that is, the edge t_1p_1, with the path $[t_1, p_2, \ldots, p_2', p_1]$. Thus, the resulting path is $P_1 = [t_1, p_2, \ldots, p_2', p_1, \ldots, p_1', t_1']$.
Note that we can replace every edge of the terminal path so that we can insert at most $|F(\{P_1\})| + 1$ non-terminal paths, where $F(\{P_1\})$ is the set of the free vertices belonging to the path P_1. If the terminal path $P_1 = [t_1, p_1, \ldots, p_1^\ell, p_1^r, \ldots, p_1', t_1']$ is constructed by connecting a semi-terminal path of S_ℓ, say, $P_\ell = [t_1, p_1, \ldots, p_1^\ell]$ with a semi-terminal path of S_r, say, $P_r = [p_1^r, \ldots, p_1', t_1']$, then it obviously has one endpoint in $G[t_\ell]$ and the other in $G[t_r]$. In this case, if $P_2 \in N_\ell$ (resp. N_r) we can only replace the edges of P_1 that belong to $G[t_r]$ (resp. $G[t_\ell]$). On the other hand, if P_2 has one endpoint, say, p_2, in N_ℓ and the other, say, p_2', in N_r, we insert P_2 into P_1 as follows: $P_1 = [t_1, p_1, \ldots, p_1^\ell, p_2', \ldots, p_2, p_1^r, \ldots, p_1', t_1']$.

We can combine the Connect and Bridge operations to perform a new operation on paths, which we call a *connect-bridge* operation; such an operation is depicted in Figure 1e and is defined below.

- *Connect-Bridge* operation: Let $P_1 = [t_1, p_1, \ldots, p_k, t_1']$ be a terminal path of the set T_ℓ (resp. T_r) and let $P_2, P_3, \ldots, P_s$ be semi-terminal paths of the set S_r (resp. S_ℓ), where s is odd and $3 \leq s \leq 2k + 3$. We say that we *connect-bridge* the

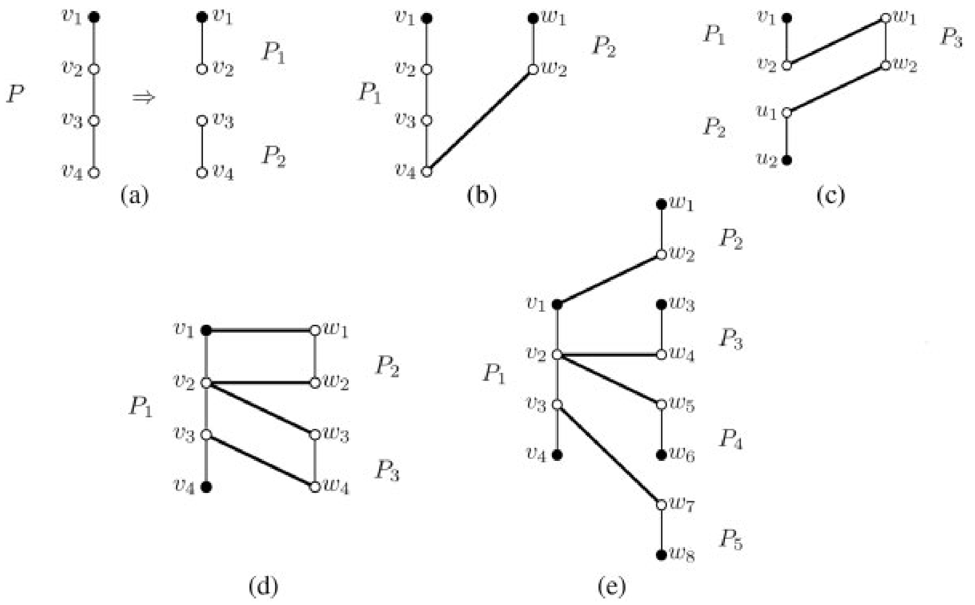

FIG. 1. Illustrating (a) break, (b) connect, (c) bridge, (d) insert, and (e) connect-bridge operations.

paths $P_2, P_3, \dots, P_s$ using vertices of P_1, if we perform the following operations:

(i) connect the path P_2 with the path $[t_1]$;
(ii) bridge $r = \frac{s-3}{2}$ pairs of semi-terminal paths using vertices $p_1, p_2, \dots, p_r$;
(iii) connect the path $[p_{r+1}, \dots, p_k, t'_1]$ with the last semi-terminal path P_s.

We point out that the Connect-Bridge operation produces two paths having one endpoint in $G[t_\ell]$ (resp. $G[t_r]$) and the other endpoint in $G[t_r]$ (resp. $G[t_\ell]$) and $\frac{s-3}{2}$ paths having both endpoints in $G[t_r]$ (resp. $G[t_\ell]$).

3. THE ALGORITHM

We next present an optimal algorithm for the kPC problem on cographs. Our algorithm takes as input a cograph G and a subset $\mathcal{T}$ of its vertices, and finds the paths of a minimum kPC of G in linear time; it works as follows:

```
Algorithm Minimum_kPC
Input: a cograph G and a set of vertices $\mathcal{T}$;
Output: a minimum kPC $\mathcal{P}_\mathcal{T}(G)$ of the cograph G;

  1. Construct the co-tree $T_{co}(G)$ of G and make it binary; let
     $T(G)$ be the resulting tree;
  2. Execute the subroutine process(root), where root is
     the root node of the tree $T(G)$; the minimum kPC
     $\mathcal{P}_\mathcal{T}(root) = \mathcal{P}_\mathcal{T}(G)$ is the set of paths returned by the
     subroutine;
```

where the description of the subroutine *process*() is as follows:

```
process (node t)
Input: node t of the modified co-tree $T(G)$ of the input graph G.
Output: a minimum kPC $\mathcal{P}_\mathcal{T}(t)$ of the cograph $G[t]$.

  1. if t is a leaf
     then return({u}), where u is the vertex associated with
     the leaf t;
     else {t is an internal node that has a left and a right
     child denoted by $t_\ell$ and $t_r$, resp.}
         $\mathcal{P}_\mathcal{T}(t_\ell) \leftarrow process(t_\ell)$;
         $\mathcal{P}_\mathcal{T}(t_r) \leftarrow process(t_r)$;
  2. if t is a P-node
     then return($\mathcal{P}_\mathcal{T}(t_\ell) \cup \mathcal{P}_\mathcal{T}(t_r)$);
  3. if t is an S-node
     then if $|N_\ell| \le |N_r|$ then swap($\mathcal{P}_\mathcal{T}(t_\ell), \mathcal{P}_\mathcal{T}(t_r)$);
         case 1: $|S_\ell| \ge |S_r|$
             call kPC_1;
         case 2: $|S_\ell| < |S_r|$
             if $|N_r| + \lfloor \frac{|S_r|-|S_\ell|}{2} \rfloor \le |F(S_\ell \cup N_\ell)|$
             then call kPC_2_a;
             else call kPC_2_b;
```

We next describe the subroutine *process*( ) in the case where t is an S-node of $T(G)$. Note that, if $|N_\ell| \le |N_r|$, we swap $\mathcal{P}_\mathcal{T}(t_\ell)$ and $\mathcal{P}_\mathcal{T}(t_r)$ and thus we have $|N_\ell| \ge |N_r|$. Consequently, we distinguish the following two cases: (1) $|S_\ell| \ge |S_r|$, and (2) $|S_\ell| < |S_r|$.

CASE 1. $|S_\ell| \ge |S_r|$

Let SN_r be the set of non-terminal paths obtained by breaking the set $S_r \cup N_r$ into $|N_\ell| - 1 + \lfloor \frac{|S_\ell|-|S_r|}{2} \rfloor$ non-terminal paths; in the case where $|N_\ell| - 1 + \lfloor \frac{|S_\ell|-|S_r|}{2} \rfloor \ge F(S_r \cup N_r)$, the paths of SN_r are trivial (recall that $F(S_r \cup N_r)$ is the set of free vertices belonging to the set $S_r \cup N_r$). The paths of SN_r are used to bridge at most $2\lfloor \frac{|S_\ell|-|S_r|}{2} \rfloor$ semi-terminal paths of S_ℓ and, if $|SN_r| - \lfloor \frac{|S_\ell|-|S_r|}{2} \rfloor > 0$, at most $|N_\ell|$ non-terminal paths of N_ℓ. Note that $|SN_r| \le |F(S_r \cup N_r)|$. We can construct the paths of a kPC using the following procedure:

Procedure kPC_1

```
  1. connect the $|S_r|$ paths of $S_r$ with $|S_r|$ paths of $S_\ell$;
  2. bridge $2\lfloor \frac{|S_\ell|-|S_r|}{2} \rfloor$ semi-terminal paths of $S_\ell$ using
     $\lfloor \frac{|S_\ell|-|S_r|}{2} \rfloor$ paths of $SN_r$;
  3. bridge the non-terminal paths of $N_\ell$ using $|N_\ell| - 1$ non-
     terminal paths of $SN_r$; this produces non-terminal paths
     with both endpoints in $G[t_\ell]$, unless $N_\ell \le |F(S_r \cup N_r)| -$
     $\lfloor \frac{|S_\ell|-|S_r|}{2} \rfloor$ where we obtain one non-terminal path with
     one endpoint in $G[t_\ell]$ and the other in $G[t_r]$;
  4. if $|N_\ell| \le |F(S_r \cup N_r)| - \lfloor \frac{|S_\ell|-|S_r|}{2} \rfloor$ insert the non-terminal
     path obtained in Step 3 into one terminal path which is
     obtained in Step 1;
  5. if $|T_r| = |S_\ell| = 0$ and $|F(S_r \cup N_r)| \ge |N_\ell|$ construct a
     non-terminal path having both of its endpoints in $G[t_r]$
     and insert it into a terminal path of $T_\ell$;
  6. if $|T_r| = |S_r| = 0$ and $|F(N_r)| \ge |N_\ell| + \lfloor \frac{|S_\ell|}{2} \rfloor$ construct
     a non-terminal path having both of its endpoints in $G[t_r]$
     and use it to connect two semi-terminal paths of $S_\ell$;
  7. if $|S_\ell| - |S_r|$ is odd and there is at least one free vertex
     in $S_r \cup N_r$ which is not used in Steps 1–4, or there is a
     non-terminal path having one endpoint in $G[t_\ell]$ and the
     other in $G[t_r]$, connect one non-terminal path with one
     semi-terminal path of $S_\ell$;
  8. connect-bridge the rest of the semi-terminal paths of $S_\ell$
     (at most $2(|F(T_r)| + |T_r|)$) using vertices of $T_r$;
  9. insert non-terminal paths obtained in Step 3 into the
     terminal paths of $T_r$;
```

On the basis of the procedure kPC_1, we can compute the cardinality of the sets N_t, S_t, and T_t, and thus, since $\lambda'_\mathcal{T}(t) = |N_t| + |S_t| + |T_t|$, the number of paths in the kPC constructed by the procedure at node $t \in T(G)$. In this case, the values of $|N_t|$, $|S_t|$ and $|T_t|$ are the following:

$$\begin{aligned}
|N_t| &= \max\{\mu - \alpha, 0\} \\
|S_t| &= \min\{\sigma_\ell, \max\{\sigma_\ell - 2(|F(T_r)| + |T_r|), \delta(\sigma_\ell)\}\} \\
|T_t| &= |S_r| + \min\left\{\left\lfloor \frac{|S_\ell| - |S_r|}{2} \right\rfloor, |F(S_r \cup N_r)|\right\} \\
&\quad + |T_\ell| + |T_r| + \frac{\sigma_\ell - |S_t|}{2} \qquad (3)
\end{aligned}$$

where

$$\sigma_\ell = |S_\ell| - |S_r| - 2\min\left\{\left\lfloor\frac{|S_\ell| - |S_r|}{2}\right\rfloor, |F(S_r \cup N_r)|\right\}$$

$$\mu = \max\{|N_\ell| - \pi_r, \max\{1 - |S_\ell|, 0\}\} - \min\left\{\max\{\min\{|N_\ell| - \pi_r, \delta(|S_\ell| - |S_r|)\}, 0\}, \max\left\{\min\left\{F(S_r \cup N_r) - \left\lfloor\frac{|S_\ell| - |S_r|}{2}\right\rfloor, 1\right\}, 0\right\}\right\} - \frac{1}{2}\max\{2(|F(T_r)| + |T_r|) - \sigma_\ell, 0\}$$

$$\alpha = \min\{\max\{\min\{\pi_r - |N_\ell|, 1\}, 0\}, \max\{|T_\ell|, 0\}\}$$

and

$$\pi_r = \max\left\{|F(S_r \cup N_r)| - \left\lceil\frac{|S_\ell| - |S_r|}{2}\right\rceil, 0\right\}.$$

In Eq. (3), σ_ℓ is the number of semi-terminal paths of S_ℓ that are not connected or bridged at Steps 1 and 2. Furthermore, π_r is the number of free vertices in the set $S_r \cup N_r$ that are not used to bridge semi-terminal paths of S_ℓ at Step 2 and δ is a function which is defined as follows: $\delta(x) = 1$, if x is odd, and $\delta(x) = 0$ otherwise. Note that at most $|F(T_r)| + |T_r|$ non-terminal paths can be inserted into the terminal paths of T_r or the terminal paths can connect-bridge at most $2(|F(T_r)| + |T_r|)$ semi-terminal paths.

CASE 2. $|S_\ell| < |S_r|$

In this case, we need $|N_r| + \lfloor\frac{|S_r|-|S_\ell|}{2}\rfloor$ paths of $G[t_\ell]$ in order to bridge $|N_r|$ non-terminal paths of N_r and $2\lfloor\frac{|S_r|-|S_\ell|}{2}\rfloor$ semi-terminal paths of S_r. If $|N_\ell| < |N_r| + \lfloor\frac{|S_r|-|S_\ell|}{2}\rfloor$ we break the non-terminal paths of N_ℓ into at most $|F(N_\ell)|$ paths; in the case where $|F(N_\ell)| < |N_r| + \lfloor\frac{|S_r|-|S_\ell|}{2}\rfloor$ we also use (at most $|F(S_\ell)|$) vertices of S_ℓ. Let $p = \min\{|N_r| + \lfloor\frac{|S_r|-|S_\ell|}{2}\rfloor, |F(S_\ell \cup N_\ell)|\}$. We distinguish two cases:

2.a $|N_r| + \lfloor\frac{|S_r|-|S_\ell|}{2}\rfloor \leq |F(S_\ell \cup N_\ell)|$.
In this case, $p = |N_r| + \lfloor\frac{|S_r|-|S_\ell|}{2}\rfloor$ and the number of non-terminal paths (or free vertices) of $G[t_\ell]$ is sufficient to bridge non-terminal paths of N_r and semi-terminal paths of S_r.

In detail, let SN_ℓ be the set of non-terminal paths obtained by breaking the set $S_\ell \cup N_\ell$ into p non-terminal paths. If $p < |N_\ell|$ then $SN_\ell = N_\ell$. The paths of SN_ℓ are used to bridge $2\lfloor\frac{|S_r|-|S_\ell|}{2}\rfloor$ semi-terminal paths of S_r and all the non-terminal paths of N_r. Obviously, $|SN_\ell| \leq |F(S_\ell \cup N_\ell)|$. Note that, if $p < |N_\ell|$ then the non-terminal paths of N_r are used to bridge the paths of N_ℓ. More precisely, we use paths of the set SN_r (it is the set of non-terminal paths that we get by breaking the set $S_r \cup N_r$) in order to obtain $|N_\ell| - \lfloor\frac{|S_r|-|S_\ell|}{2}\rfloor$ non-terminal paths. If $p \geq |N_\ell|$ we set $SN_r = N_r$ and we use at most $|F(S_\ell \cup N_\ell)|$ paths obtained by $S_\ell \cup N_\ell$ in order to bridge non-terminal paths of N_r and semi-terminal paths of S_r, that is, we use the set SN_ℓ. As a result, we construct $\lfloor\frac{|S_r|-|S_\ell|}{2}\rfloor$ terminal paths having both of their endpoints in $G[t_r]$ and we have at least one non-terminal path, if $p < |N_\ell|$, and exactly one non-terminal path, otherwise. Note that, in the second case, we can construct the non-terminal path in such a way that one endpoint is in SN_ℓ and the other is in N_r. Consequently, we can connect the path to a semi-terminal path of S_r, in the case where $|S_r| - |S_\ell|$ is an odd number, or we can insert it into a terminal path which is obtained by connecting $|S_\ell|$ paths of S_ℓ with $|S_\ell|$ paths of S_r. We construct the paths of a kPC at node $t \in T(G)$ using the following procedure:

Procedure kPC_2_a

1. connect the $|S_\ell|$ paths of S_ℓ with $|S_\ell|$ paths of S_r;
2. if $|T_\ell| = |T_r| = 0$ and $p \geq |N_\ell|$, use N_r to bridge $p - \lfloor\frac{|S_r|-|S_\ell|}{2}\rfloor + 1$ paths of SN_ℓ and use the constructed non-terminal path having both of its endpoints in $G[t_\ell]$ to bridge two semi-terminal paths of S_r;
3. bridge semi-terminal paths of S_r using paths of SN_ℓ;
4. if $|T_r| = 0, |T_\ell| \neq 0$, $p \geq |N_\ell|$ and $|F(S_r \cup N_r)| \geq |SN_\ell| - \lfloor\frac{|S_r|-|S_\ell|}{2}\rfloor$ construct a non-terminal path having both of its endpoints in $G[t_r]$ and insert into a terminal path of T_ℓ;
5. bridge the remaining paths of SN_ℓ using the paths of SN_r. This produces non-terminal paths one of which has one endpoint in $G[t_\ell]$ and the other in $G[t_r]$;
6. if $|S_r| - |S_\ell|$ is odd, we connect one non-terminal path with one semi-terminal path of S_r;
7. insert at most $|F(T_r)| + |T_r|$ non-terminal paths obtained in Step 5 into the terminal paths of T_r;

Based on the path operations performed by procedure kPC_2_a, we can show that the sets N_t, S_t and T_t, have the following cardinalities:

$$|N_t| = \max\{\mu - \alpha, 0\}$$
$$|S_t| = \delta(|S_r| - |S_\ell|)$$
$$|T_t| = |S_\ell| + \left\lfloor\frac{|S_r| - |S_\ell|}{2}\right\rfloor + |T_\ell| + |T_r| \tag{4}$$

where

$$\mu = \max\{|N_\ell| - F(S_r \cup N_r), 0\} - \left\lfloor\frac{|S_r| - |S_\ell|}{2}\right\rfloor - \delta(|S_r| - |S_\ell|) - |F(T_r)| - |T_r|$$

$$\alpha = \min\{\max\{\min\{F(S_r \cup N_r) - |N_\ell|, 1\}, 0\}, \max\{|T_\ell|, 0\}\}$$

Recall that $\delta(x) = 1$, if x is odd, and $\delta(x) = 0$ otherwise.

2.b $|N_r| + \lfloor\frac{|S_r|-|S_\ell|}{2}\rfloor > |F(S_\ell \cup N_\ell)|$.
In this case, $p = |F(S_\ell \cup N_\ell)|$ and the number of non-terminal paths (or free vertices) of $G[t_\ell]$ (either in N_ℓ or in S_ℓ) is not sufficient to bridge non-terminal paths of N_r and semi-terminal paths of S_r.

In detail, let SN_ℓ be the set of the trivial, non-terminal paths, obtained by breaking the set $S_\ell \cup N_\ell$ into $|F(S_\ell \cup N_\ell)|$ non-terminal paths. Note that $SN_\ell = F(S_\ell \cup N_\ell)$. Similar to Case 1, we can construct the paths of a kPC at node $t \in T(G)$ using the following procedure:

Procedure kPC_2_b

1. connect the $|S_\ell|$ paths of S_ℓ with $|S_\ell|$ paths of S_r;
2. bridge $2\lfloor\frac{|S_r|-|S_\ell|}{2}\rfloor$ semi-terminal paths of S_r using $\lfloor\frac{|S_r|-|S_\ell|}{2}\rfloor$ paths of SN_ℓ;
3. bridge the non-terminal paths of N_r using the rest of the non-terminal paths of SN_ℓ. This produces non-terminal paths such that both endpoints belong to $G[t_r]$;
4. connect-bridge the rest of the semi-terminal paths of S_r (at most $2(|F(T_\ell)| + |T_\ell|)$) using vertices of T_ℓ;
5. insert non-terminal paths obtained in Step 3 into the terminal paths of T_ℓ;

After computing the number of non-terminal, semi-terminal and terminal paths produced by procedure kPC_2_b, we conclude that:

$$\begin{aligned}
|N_t| &= \max\{\mu, 0\}\\
|S_t| &= \min\{\sigma_r, \max\{\sigma_r - 2(|F(T_\ell)| + |T_\ell|), \delta(\sigma_r)\}\}\\
|T_t| &= |S_\ell| + \min\left\{\left\lfloor\frac{|S_r|-|S_\ell|}{2}\right\rfloor, |F(S_\ell \cup N_\ell)|\right\}\\
&\quad + |T_\ell| + |T_r| + \frac{\sigma_r - |S_t|}{2} \qquad (5)
\end{aligned}$$

where

$$\begin{aligned}
\sigma_r &= |S_r| - |S_\ell| - 2\min\left\{\left\lfloor\frac{|S_r|-|S_\ell|}{2}\right\rfloor, |F(S_\ell \cup N_\ell)|\right\}\\
\mu \ &= |N_r| - \pi_\ell - \min\left\{\max\left\{\min\left\{|F(S_\ell \cup N_\ell)|\right.\right.\right.\\
&\qquad \left.\left. - \left\lfloor\frac{|S_r|-|S_\ell|}{2}\right\rfloor, 1\right\}, 0\right\},\\
&\qquad \left.\max\{\min\{|N_r| - \pi_\ell, \delta(|S_r| - |S_\ell|)\}, 0\}\right\}\\
&\quad - \frac{1}{2}\max\{2(|F(T_\ell)| + |T_\ell|) - \sigma_r, 0\}
\end{aligned}$$

and

$$\pi_\ell = \max\left\{|F(S_\ell \cup N_\ell)| - \left\lceil\frac{|S_r|-|S_\ell|}{2}\right\rceil, 0\right\}.$$

In Equation (5), σ_r is the number of semi-terminal paths of S_r that are not connected or bridged at Steps 1 and 2. Moreover, π_ℓ is the number of free vertices that belong to the set $S_\ell \cup N_\ell$ and are not used to bridge semi-terminal paths of S_r (at Step 2). Again, $\delta(x) = 1$, if x is odd, and $\delta(x) = 0$ otherwise. Note that at most $|F(T_\ell)| + |T_\ell|$ non-terminal paths can be inserted into the terminal paths of T_ℓ or the terminal paths can connect-bridge at most $2(|F(T_\ell)| + |T_\ell|)$ semi-terminal paths.

4. CORRECTNESS AND TIME COMPLEXITY

Let G be a cograph, $T(G)$ be the modified co-tree of G, and let $\mathcal{T}$ be a terminal set of G. Since our algorithm computes a kPC $\mathcal{P}'_\mathcal{T}(t)$ of $G[t]$ of size $\lambda'_\mathcal{T}(t)$ for each internal node $t \in T(G)$, and thus for the root $t = t_{root}$ of the tree $T(G)$, we need to prove that the constructed kPC $\mathcal{P}'_\mathcal{T}(t)$ is minimum. Obviously, the size $\lambda_\mathcal{T}(t)$ of a minimum kPC of the graph $G[t]$ is less than or equal to the size $\lambda'_\mathcal{T}(t)$ of the kPC constructed by our algorithm. According to Proposition 2.1, if the size of the kPC constructed by our algorithm is $\lambda'_\mathcal{T}(t) = \lceil\frac{|\mathcal{T}_t|}{2}\rceil$, then it is a minimum kPC. After performing simple computations on Eqs. (3)–(5), we get four specific values for the size $\lambda'_\mathcal{T}(t)$ of the path cover constructed by our algorithm, that is, by the kPC procedures 1, 2_a, and 2_b. More precisely, our algorithm returns a kPC of size $\lambda'_\mathcal{T}(t)$ equal to either $\frac{|\mathcal{T}_t|}{2} + 1$, $\lceil\frac{|\mathcal{T}_t|}{2}\rceil$, $\lambda_\mathcal{T}(t_\ell) - |F(V_r)|$, or $\lambda_\mathcal{T}(t_r) - |F(V_\ell)|$; see Table 1. Recall that $\mathcal{T}_t$ denotes the set of terminal vertices in $V_t = V(G[t])$, while $F(V_\ell)$ (resp. $F(V_r)$) denotes the set of free vertices in $V_\ell = V(G[t_\ell])$ (resp. $V_r = V(G[t_r])$); t_ℓ and t_r are the left child and right child, respectively, of node t.

Let t be an internal node of $T(G)$ and let N_ℓ, S_ℓ and T_ℓ (resp. N_r, S_r and T_r) be the sets of non-terminal, semi-terminal and terminal paths, respectively, in $G[t_\ell]$ (resp. $G[t_r]$). For the case where $|S_\ell| = |T_r| = |S_r| = 0$ and $|N_\ell| = |V_r|$ our algorithm (procedure kPC_1) returns a kPC of the graph $G[t]$ of size $\lambda'_\mathcal{T}(t) = \frac{|\mathcal{T}_t|}{2} + 1$. We prove the following lemma:

Lemma 4.1. *Let t be an S-node of $T(G)$ and let $\mathcal{P}_\mathcal{T}(t_\ell)$ and $\mathcal{P}_\mathcal{T}(t_r)$ be a minimum kPC of $G[t_\ell]$ and $G[t_r]$, respectively. If $|S_\ell| = |T_r| = |S_r| = 0$ and $|N_\ell| = |V_r|$, then the procedure kPC_1 returns a minimum kPC of $G[t]$ of size $\lambda_\mathcal{T}(t) = \frac{|\mathcal{T}_t|}{2} + 1$.*

Proof. Since we can construct a kPC of size $\lambda'_\mathcal{T}(t) = \frac{|\mathcal{T}_t|}{2} + 1$, then the size $\lambda_\mathcal{T}(t)$ of a minimum kPC is at most $\frac{|\mathcal{T}_t|}{2} + 1$. We will show that we can not construct a minimum kPC of size less than $\frac{|\mathcal{T}_t|}{2} + 1$, that is, we will show that $\lambda_\mathcal{T}(t) \geq \frac{|\mathcal{T}_t|}{2} + 1 \Leftrightarrow \lambda_\mathcal{T}(t) > \frac{|\mathcal{T}_t|}{2}$. Thus, we only need to prove that $\lambda_\mathcal{T}(t) \neq \frac{|\mathcal{T}_t|}{2}$. Note that by the assumption we have $\frac{|\mathcal{T}_t|}{2} = |T_\ell|$. We assume that $\lambda_\mathcal{T}(t) = \frac{|\mathcal{T}_t|}{2}$, and, thus, $\lambda_\mathcal{T}(t) = |T_\ell|$. There exists at least one non-terminal path in $G[t_\ell]$; for otherwise $|N_\ell| = 0$, and thus $V_r = \emptyset$, a contradiction. We ignore the terminal paths from the minimum kPC of $G[t_\ell]$ and apply the algorithm described in [18] to $G[t]$. The resulting minimum kPC contains only one (non-terminal) path which either has both endpoints in $G[t_\ell]$ or it has one endpoint in $G[t_\ell]$ and the other in $G[t_r]$. This non-terminal path can not be inserted into a terminal path of $G[t_\ell]$ because it does not have both endpoints in $G[t_r]$. Thus, $\lambda_\mathcal{T}(t) = |T_\ell| + 1$, a contradiction. ∎

TABLE 1. The size of the kPC that our algorithm returns in each case.

Procedures	Size of k-fixed-endpoint PC
Procedure kPC_1	$\frac{|\mathcal{T}_t|}{2} + 1$
All the procedures	$\lceil\frac{|\mathcal{T}_t|}{2}\rceil$
Procedure kPC_1 and Procedure kPC_2_a	$\lambda_\mathcal{T}(t_\ell) - |F(V_r)|$
Procedure kPC_2_b	$\lambda_\mathcal{T}(t_r) - |F(V_\ell)|$

The previous lemma shows that if the size of the kPC returned by our subroutine process(t) for the graph $G[t]$ is $\lambda'_{\mathcal{T}}(t) = \frac{|\mathcal{T}_t|}{2} + 1$ (procedure kPC_1), then it is a minimum kPC. Moreover, if the size of the kPC returned by the process(t) is $\lceil\frac{|\mathcal{T}_t|}{2}\rceil$ (all the procedures), then it is obviously a minimum kPC of $G[t]$. We will prove that the size $\lambda'_{\mathcal{T}}(t)$ of the kPC $\mathcal{P}'_{\mathcal{T}}(t)$ that our subroutine process(t) returns is minimum.

Lemma 4.2. *Let t be an S-node of $T(G)$ and let $\mathcal{P}_{\mathcal{T}}(t_\ell)$ and $\mathcal{P}_{\mathcal{T}}(t_r)$ be a minimum kPC of $G[t_\ell]$ and $G[t_r]$, respectively. If the subroutine process(t) returns a kPC of $G[t]$ of size $\lambda'_{\mathcal{T}}(t) = \lceil\frac{|\mathcal{T}_t|}{2}\rceil$, then $\lambda'_{\mathcal{T}}(t) \geq \max\{\lambda_{\mathcal{T}}(t_\ell) - |F(V_r)|, \lambda_{\mathcal{T}}(t_r) - |F(V_\ell)|\}$.*

Proof. Since $\lambda'_{\mathcal{T}}(t) = \lceil\frac{|\mathcal{T}_t|}{2}\rceil$, we have $\lambda'_{\mathcal{T}}(t) = \lambda_{\mathcal{T}}(t)$, that is, the kPC that the subroutine process(t) returns is minimum. Thus, the proof follows from Lemma 2.1. ■

Let t be an S-node of $T(G)$ and let $\mathcal{P}_{\mathcal{T}}(t_\ell)$ and $\mathcal{P}_{\mathcal{T}}(t_r)$ be a minimum kPC of $G[t_\ell]$ and $G[t_r]$, respectively. Furthermore, we assume that the conditions $|S_\ell| = |T_r| = |S_r| = 0$ and $|N_\ell| = |V_r|$ do not hold together. We consider the case where the subroutine process(t) returns a kPC $\mathcal{P}'_{\mathcal{T}}(t)$ of the graph $G[t]$ of size $\lambda'_{\mathcal{T}}(t) = \lambda_{\mathcal{T}}(t_\ell) - |F(V_r)|$ (cases 1 and 2.a). We prove the following lemma.

Lemma 4.3. *Let t be an S-node of $T(G)$ and let $\mathcal{P}_{\mathcal{T}}(t_\ell)$ and $\mathcal{P}_{\mathcal{T}}(t_r)$ be a minimum kPC of $G[t_\ell]$ and $G[t_r]$, respectively. If the subroutine process(t) returns a kPC of $G[t]$ of size $\lambda'_{\mathcal{T}}(t) = \lambda_{\mathcal{T}}(t_\ell) - |F(V_r)|$, then $\lambda'_{\mathcal{T}}(t) > \max\{\lceil\frac{|\mathcal{T}_t|}{2}\rceil, \lambda_{\mathcal{T}}(t_r) - |F(V_\ell)|\}$.*

Proof. We consider the cases 1 and 2.a. In these cases, the size $\lambda'_{\mathcal{T}}(t)$ of the constructed kPC is computed using Equations (3) and (4) and the fact that $\lambda'_{\mathcal{T}}(t) = |N_t| + |S_t| + |T_t|$. After performing simple computations, we conclude that in these cases the subroutine process(t) returns $\lambda'_{\mathcal{T}}(t) = \lambda_{\mathcal{T}}(t_\ell) - |F(V_r)|$ if the following condition holds:

$$|N_\ell| + \left\lceil\frac{|S_\ell| - |S_r|}{2}\right\rceil > |F(V_r)| + |T_r| + \frac{\delta(|S_\ell| - |S_r|)}{2}. \tag{6}$$

We will show that (i) $\lambda'_{\mathcal{T}}(t) > \lceil\frac{|\mathcal{T}_t|}{2}\rceil$ and, (ii) $\lambda'_{\mathcal{T}}(t) > \lambda_{\mathcal{T}}(t_r) - |F(V_\ell)|$. We first consider the case where $\delta(|S_\ell| - |S_r|) = 0$. In this case either the values of $|S_\ell|$ and $|S_r|$ are both odd numbers or they are both even numbers. In any case, $|S_\ell| + |S_r|$ is an even number and thus, $|\mathcal{T}_t|$ is also an even number. Thus, according to Proposition 2.1 and since $G[t]$ is a cograph, we have $|\mathcal{T}_t| = |S_\ell| + |S_r| + 2|T_\ell| + 2|T_r|$.
(i) We first show that $\lambda'_{\mathcal{T}}(t) = \lambda_{\mathcal{T}}(t_\ell) - |F(V_r)| > \lceil\frac{|\mathcal{T}_t|}{2}\rceil$. From Eq. (6) and, since $2|T_r| + |S_r| = |\mathcal{T}_r|$ (see Proposition 2.1), we obtain

$$2|F(V_r)| + |\mathcal{T}_r| < 2|N_\ell| + |S_\ell|. \tag{7}$$

By Proposition 2.1 and Eq. (7) we have

$$\begin{aligned}&2(|N_\ell| + |S_\ell| + |T_\ell| - |F(V_r)|)\\ &> 2|F(V_r)| + |\mathcal{T}_r| + |S_\ell| + 2|T_\ell| - 2|F(V_r)| = |\mathcal{T}_\ell| + |\mathcal{T}_r| = |\mathcal{T}_t|.\end{aligned}$$

Since $\lambda_{\mathcal{T}}(t_\ell) - |F(V_r)| = |N_\ell| + |S_\ell| + |T_\ell| - |F(V_r)|$, it follows that $\lambda_{\mathcal{T}}(t_\ell) - |F(V_r)| > \lceil\frac{|\mathcal{T}_t|}{2}\rceil$.
(ii) We next show that $\lambda'_{\mathcal{T}}(t) = \lambda_{\mathcal{T}}(t_\ell) - |F(V_r)| > \lambda_{\mathcal{T}}(t_r) - |F(V_\ell)|$. According to Proposition 2.2, we have $\lambda_{\mathcal{T}}(t_r) - |F(V_\ell)| \leq |F(V_r)| + |\mathcal{T}_r| - |F(V_\ell)|$. From Eq. (7) and since

$$|N_\ell| \leq |F(N_\ell)| \Leftrightarrow |N_\ell| \leq |F(V_\ell)| \Leftrightarrow |N_\ell| - |F(V_\ell)| \leq 0,$$

we obtain $|F(V_r)| + |\mathcal{T}_r| - |F(V_\ell)| < 2|N_\ell| + |S_\ell| - |F(V_r)| - |F(V_\ell)| \leq 2|N_\ell| + |S_\ell| + |T_\ell| - |F(V_r)| - |F(V_\ell)| = \lambda_{\mathcal{T}}(t_\ell) - |F(V_r)| + |N_\ell| - |F(V_\ell)| \leq \lambda_{\mathcal{T}}(t_\ell) - |F(V_r)|$.

Similarly, we can prove that $\lambda_{\mathcal{T}}(t_\ell) - |F(V_r)| > \lceil\frac{|\mathcal{T}_t|}{2}\rceil$ and $\lambda_{\mathcal{T}}(t_\ell) - |F(V_r)| > \lambda_{\mathcal{T}}(t_r) - |F(V_\ell)|$, for the case where $\delta(|S_\ell| - |S_r|) = 1$, and, thus, $\lambda_{\mathcal{T}}(t_\ell) - |F(V_r)| > \max\{\lceil\frac{|\mathcal{T}_t|}{2}\rceil, \lambda_{\mathcal{T}}(t_r) - |F(V_\ell)|\}$. ■

Using arguments similar to those used to prove Lemma 4.3, we can show that if the subroutine process(t) returns a kPC of $G[t]$ of size $\lambda'_{\mathcal{T}}(t) = \lambda_{\mathcal{T}}(t_r) - |F(V_\ell)|$ (case 2.b), then $\lambda'_{\mathcal{T}}(t) > \max\{\lceil\frac{|\mathcal{T}_t|}{2}\rceil, \lambda_{\mathcal{T}}(t_\ell) - |F(V_r)|\}$; note that, in this case we have $|N_r| + \lceil\frac{|S_r| - |S_\ell|}{2}\rceil > |F(V_\ell)| + |T_\ell| + \frac{\delta(|S_r| - |S_\ell|)}{2}$. Thus, we have proved the following result.

Lemma 4.4. *Let t be an S-node of $T(G)$ and let $\mathcal{P}_{\mathcal{T}}(t_\ell)$ and $\mathcal{P}_{\mathcal{T}}(t_r)$ be a minimum kPC of $G[t_\ell]$ and $G[t_r]$, respectively. The subroutine process(t) returns a kPC $\mathcal{P}_{\mathcal{T}}(t)$ of $G[t]$ of size*

$$\lambda'_{\mathcal{T}}(t) = \begin{cases} \frac{|\mathcal{T}_t|}{2} + 1 & \text{if } |S_\ell| = |\mathcal{T}_r| = 0 \text{ and } \\ & \quad |N_\ell| = |V_r|, \\ \max\Big\{\left\lceil\frac{|\mathcal{T}_t|}{2}\right\rceil, & \\ \quad \lambda_{\mathcal{T}}(t_\ell) - |F(V_r)|, & \\ \quad \lambda_{\mathcal{T}}(t_r) - |F(V_\ell)|\Big\} & \text{otherwise.} \end{cases}$$

Obviously, a minimum kPC of the graph $G[t]$ is of size $\lambda_{\mathcal{T}}(t) \leq \lambda'_{\mathcal{T}}(t)$. On the other hand, we have proved a lower bound for the size $\lambda_{\mathcal{T}}(t)$ of a minimum kPC of the graph $G[t]$ (see Lemma 2.1), namely, $\lambda_{\mathcal{T}}(t) \geq \max\{\lceil\frac{|\mathcal{T}_t|}{2}\rceil, \lambda_{\mathcal{T}}(t_\ell) - |F(V_r)|, \lambda_{\mathcal{T}}(t_r) - |F(V_\ell)|\}$. It follows that $\lambda'_{\mathcal{T}}(t) = \lambda_{\mathcal{T}}(t)$, and, thus, we can state the following result.

Lemma 4.5. *Subroutine process(t) returns a minimum kPC $\mathcal{P}_{\mathcal{T}}(t)$ of the graph $G[t]$, for every internal S-node $t \in T(G)$.*

Since the above result holds for every S-node t of the modified co-tree $T(G)$, it also holds when t is the root of $T(G)$ and $\mathcal{T}_t = \mathcal{T}$. Thus, the following theorem holds:

Theorem 4.1. *Let G be a cograph and let $\mathcal{T}$ be a terminal set of G. Let t be the root of the modified co-tree $T(G)$, and* *let $\mathcal{P}_{\mathcal{T}}(t_\ell)$ and $\mathcal{P}_{\mathcal{T}}(t_r)$ be a minimum kPC of $G[t_\ell]$ and $G[t_r]$, respectively. Algorithm Minimum_kPC correctly computes a minimum kPC of $G = G[t]$ with respect to $\mathcal{T} = \mathcal{T}_t$ of size $\lambda_{\mathcal{T}} = \lambda_{\mathcal{T}}(t)$, where*

$$\lambda_{\mathcal{T}}(t) = \begin{cases} \lambda_{\mathcal{T}}(t_r) + \lambda_{\mathcal{T}}(t_\ell) & \text{if } t \text{ is a P-node,} \\ \frac{|\mathcal{T}_t|}{2} + 1 & \text{if } t \text{ is an S-node and} \\ & |S_\ell| = |\mathcal{T}_r| = 0 \text{ and} \\ & \quad |N_\ell| = |V_r|, \\ \max\left\{\left\lceil \frac{|\mathcal{T}_t|}{2} \right\rceil, \right. & \\ \quad \lambda_{\mathcal{T}}(t_\ell) - |F(V_r)|, & \\ \quad \left. \lambda_{\mathcal{T}}(t_r) - |F(V_\ell)| \right\} & \text{otherwise.} \end{cases}$$

Let G be a cograph on n vertices and m edges, $\mathcal{T}$ be a terminal set, and let t be an S-node of the modified co-tree $T(G)$. From the description of the algorithm we can easily conclude that a minimum kPC $\mathcal{P}_{\mathcal{T}}(t)$ of $G[t]$ can be constructed in $O(E(G[t]))$ time, since we use at most $|V(G[t_\ell])| \cdot |V(G[t_r])|$ edges to connect the paths of the minimum kPCs of the graphs $G[t_\ell]$ and $G[t_r]$; in the case where t is a P-node a minimum kPC is constructed in $O(1)$ time. Thus, the time needed by the subroutine process(t) to compute a minimum kPC in the case where t is the root of the tree $T(G)$ is $O(n + m)$; moreover, through the execution of the subroutine no additional space is needed. The construction of the co-tree $T_{\text{co}}(G)$ of G needs $O(n + m)$ time and it requires $O(n)$ space [4, 6]. Furthermore, the binarization process of the co-tree, that is, the construction of the modified co-tree $T(G)$, takes $O(n)$ time. Hence, we can state the following result.

Theorem 4.2. *Let G be a cograph on n vertices and m edges and let $\mathcal{T}$ be a terminal set of G. A minimum k-fixed-endpoint path cover $\mathcal{P}_{\mathcal{T}}$ of G can be computed in $O(n + m)$ time and space.*

5. CONCLUDING REMARKS

This paper presents a simple linear-time algorithm for the k-fixed-endpoint path cover problem on cographs. Given a cograph G and a set of vertices $\mathcal{T}$, our algorithm constructs a minimum k-fixed-endpoint path cover of G that contains a large number of terminal paths.

Thus, it is worth investigating the existence of a linear-time algorithm for finding a minimum k-fixed-endpoint path cover on cographs such that it contains a large number of semi-terminal paths; we pose it as an open problem.

It would also be interesting to define a variant of the k-fixed-endpoint path cover problem, which would include a pair of terminal sets $\mathcal{T}_1$ and $\mathcal{T}_2$. In this variant, we want to compute a minimum path cover such that all the terminal paths have one endpoint in $\mathcal{T}_1$ and the other in $\mathcal{T}_2$. Then, we could use this extended k-fixed-endpoint path cover problem to solve problem B (see Section 1).

Finally, an interesting open question would also be to see if the k-fixed-endpoint path cover problem can be polynomially solved on other classes of graphs; an interesting next step would be to consider the class of interval graphs. This promises to be an interesting area for further research.

REFERENCES

[1] G.S. Adhar and S. Peng, Parallel algorithm for path covering, Hamiltonian path, and Hamiltonian cycle in cographs, International Conference on Parallel Processing, Vol. III, Algorithms and Architecture, Pennsylvania State University Press, PA, 1990, pp. 364–365.

[2] S.R. Arikati and C.P. Rangan, Linear algorithm for optimal path cover problem on interval graphs, Inf Process Lett 35 (1990), 149–153.

[3] A. Brandstädt, V.B. Le, and J. Spinrad, Graph classes—A survey, SIAM Monographs in Discrete Mathematics and Applications, SIAM, Philadelphia, 1999.

[4] D.G. Corneil, H. Lerchs, and L. Stewart Burlingham, Complement reducible graphs, Discr Appl Math 3 (1981), 163–174.

[5] D.G. Corneil and Y. Perl, Clustering and domination in perfect graphs, Discr Appl Math 9 (1984), 27–39.

[6] D.G. Corneil, Y. Perl, and L.K. Stewart, A linear recognition algorithm for cographs, SIAM J Comput 14 (1985), 926–984.

[7] P. Damaschke, Paths in interval graphs and circular arc graphs, Discrete Math 112 (1993), 49–64.

[8] M.R. Garey and D.S. Johnson, Computers and intractability: A guide to the theory of NP-completeness, W. H. Freeman, San Francisco, 1979.

[9] V. Giakoumakis, F. Roussel, and H. Thuillier, On P_4-tidy graphs, Discrete Math Theor Comput Sci 1 (1997), 17–41.

[10] C. Hoáng, Perfect graphs, Ph.D. Thesis, McGill University, Montreal, Canada, 1985.

[11] W. Hochstättler and G. Tinhofer, Hamiltonicity in graphs with few P_4's, Computing 54 (1995), 213–225.

[12] S.Y. Hsieh, An efficient parallel strategy for the two-fixed-endpoint Hamiltonian path problem on distance-hereditary graphs, J Parallel Distrib Comput 64 (2004), 662–685.

[13] S.Y. Hsieh, C.W. Ho, T.S. Hsu, and M.T. Ko, The Hamiltonian problem on distance-hereditary graphs, Discr Appl Math 154 (2006), 508–524.

[14] R.W. Hung and M.S. Chang, Linear-time algorithms for the Hamiltonian problems on distance-hereditary graphs, Theor Comp Sci 341 (2005), 411–440.

[15] R.W. Hung and M.S. Chang, Solving the path cover problem on circular-arc graphs by using an approximation algorithm, Discr Appl Math 154 (2006), 76–105.

[16] H.A. Jung, On a class of posets and the corresponding comparability graphs, J Comb Theory (B) 24 (1978), 125–133.

[17] H. Lerchs, On cliques and kernels, Technical Report, Department of Computer Science, University of Toronto, 1971.

[18] R. Lin, S. Olariu, and G. Pruesse, An optimal path cover algorithm for cographs, Comput Math Appl 30 (1995), 75–83.

[19] R.M. McConnell and J. Spinrad, Modular decomposition and transitive orientation, Discrete Math 201 (1999), 189–241.

[20] K. Nakano, S. Olariu, and A.Y. Zomaya, A time-optimal solution for the path cover problem on cographs, Theor Comp Sci 290 (2003), 1541–1556.

[21] S.D. Nikolopoulos, Parallel algorithms for Hamiltonian problems on quasi-threshold graphs, J Parallel Distrib Comput 64 (2004), 48–67.

[22] R. Srikant, R. Sundaram, K.S. Singh, and C.P. Rangan, Optimal path cover problem on block graphs and bipartite permutation graphs, Theor Comp Sci 115 (1993), 351–357.

[23] Y. Suzuki, K. Kaneko, and M. Nakamori, Node-disjoint paths algorithm in a transposition graph, IEICE Trans Inf Syst E89-D (2006), 2600–2605.