Algorithms for Finding Disjoint Path Covers in Unit Interval Graphs

Jung-Heum Park\textsuperscript{a}, Joonsoo Choi\textsuperscript{b}, Hyeong-Seok Lim\textsuperscript{c,*}

\textsuperscript{a}School of Computer Science and Information Engineering, The Catholic University of Korea, Korea
\textsuperscript{b}School of Computer Science, Kookmin University, Korea
\textsuperscript{c}School of Electronics and Computer Engineering, Chonnam National University, Korea

Abstract
A many-to-many \textit{k}-disjoint path cover (\textit{k}-DPC for short) of a graph \(G\) joining the pairwise disjoint vertex sets \(S\) and \(T\), each of size \(k\), is a collection of \(k\) vertex-disjoint paths between \(S\) and \(T\), which altogether cover every vertex of \(G\). This is classified as paired, if each vertex of \(S\) must be joined to a specific vertex of \(T\), or unpaired, if there is no such constraint. In this paper, we develop a linear-time algorithm for the Unpaired DPC problem of finding an unpaired DPC joining \(S\) and \(T\) given in a unit interval graph, to which the One-to-One DPC and the One-to-Many DPC problems are reduced in linear time. Additionally, we show that the Paired \(k\)-DPC problem on a unit interval graph is polynomially solvable for any fixed \(k\).

Keywords: Disjoint path; path cover; path partition; proper interval graph; graph algorithm.

1. Introduction

Let \(G\) be a simple undirected graph, whose vertex and edge sets are denoted by \(V(G)\) and \(E(G)\), respectively. A path cover of graph \(G\) is a set of paths that altogether cover every vertex of \(G\). Of special interest is the vertex-disjoint path cover, or simply called disjoint path cover, which has the following additional constraint: every vertex must belong to one and only one path. The disjoint path cover problem finds applications in many areas, such as software testing, database design, and code optimization [2, 28]. In addition, the problem is concerned with applications where full utilization of network nodes is important [32].

The original disjoint path cover problem has no constraints on the positions of terminals or on the lengths of paths. The problem is to determine a disjoint path cover of a graph that uses the minimum number of paths. The minimum number is said to be the path cover number of the graph. The path cover (number) problem for a general graph is NP-complete [15], because the path cover number is equal to one if and only if the graph contains a hamiltonian path. Polynomial-time algorithms have been developed for some classes of graphs, for example, interval graphs [1], block graphs and bipartite permutation graphs [39], cographs [25], and distance-hereditary graphs [13].

\textsuperscript{*}Corresponding author

Email addresses: j.h.park@catholic.ac.kr (Jung-Heum Park), jschoi@kookmin.ac.kr (Joonsoo Choi), hslim@chonnam.ac.kr (Hyeong-Seok Lim)
In this paper, we are concerned with the disjoint path cover problem with prescribed sources and sinks, where each path should run from a source to a sink. The disjoint path cover made of \( k \) paths is called the \( k \)-disjoint path cover (\( k \)-DPC for short). Given two pairwise disjoint terminal sets, a source set \( S = \{s_1, \ldots, s_k\} \) and a sink set \( T = \{t_1, \ldots, t_k\} \), of graph \( G \), the many-to-many \( k \)-DPC is a disjoint path cover, each of whose paths joins a pair of source and sink. The disjoint path cover is paired if every source \( s_i \) must be matched with a specific sink \( t_i \). On the other hand, it is unpaired if any permutation of sinks may be mapped bijectively to sources. There are two simpler variants: the one-to-many \( k \)-DPC for \( S = \{s\} \) and \( T = \{t_1, \ldots, t_k\} \), whose paths join the common source to \( k \) distinct sinks; and the one-to-one \( k \)-DPC for \( S = \{s\} \) and \( T = \{t\} \), whose paths always start from the common source and end up in the common sink. The disjoint path covers of this type have been studied for graphs, such as hypercubes \([7, 8, 13, 17, 20]\), recursive circulants \([21, 22]\), hypercube-like graphs \([19, 23, 32, 33]\), \( k \)-ary \( n \)-cubes \([37, 40]\), cubes of connected graphs \([29, 30]\), and grid graphs \([31]\).

Some other types of the disjoint path cover problem can also be found in the literature. Given a set of \( k \) sources, \( S = \{s_1, \ldots, s_k\} \), in graph \( G \), which is associated with \( k \) positive integers, \( l_1, \ldots, l_k \), such that \( \sum_{i=1}^{k} l_i = |V(G)| \), a prescribed-source-and-length \( k \)-DPC of \( G \) is a disjoint path cover composed of \( k \) paths, each of whose paths starts at source \( s_i \) and contains \( l_i \) vertices for \( i \in \{1, \ldots, k\} \). For studies on this type of DPCs, refer to \([10, 27]\). Given a graph \( G \) and a subset \( T \) of \( k \) vertices of \( G \), a \( k \)-fixed endpoint path cover of \( G \) with respect to \( T \) is a set of vertex-disjoint paths that covers the vertices of \( G \), such that the \( k \) vertices of \( T \) are all terminals of the paths in the DPC. For details, refer to \([2, 3]\).

An interval graph is the intersection graph of family \( I \) of intervals on the real line, where two vertices are connected with an edge if and only if their corresponding intervals intersect. The family \( I \) is usually called an interval representation for the graph. A unit interval graph is an interval graph with an interval representation in which all the intervals have unit length. Refer to Fig. 1 for an example of a unit interval graph and its interval representation. In a similar way, a proper interval graph is an interval graph with an interval representation in which no interval properly contains another. In 1969, Roberts \([34]\) proved that the classes of unit interval graphs and proper interval graphs coincide.

Fig. 1: A unit interval graph and its interval representation.
An ordering, \((v_1, \ldots, v_n)\), of the vertices of a graph of order \(n\) is \textit{consecutive} if the vertices contained in a maximal clique are consecutive. A unit interval graph always admits a consecutive ordering because it is evident that the sequence of unit intervals sorted by their left endpoints corresponds to a consecutive order. See Fig. 1 again, where as well as \((v_1, \ldots, v_{17})\), the ordering \((v_1, \ldots, v_{15}, v_{17}, v_{16})\) with \(v_{16}\) and \(v_{17}\) being switched is also consecutive. A unit interval representation and a consecutive ordering of a unit interval graph can be computed in time linear to the size of the graph \([11][12]\). The class of the unit interval graphs is known to admit polynomial solutions for many problems that are NP-complete for general graphs, such as vertex coloring, clique, independent set, etc. \([16]\).

Given a source set \(S = \{s_1, \ldots, s_k\}\) and a sink set \(T = \{t_1, \ldots, t_k\}\) in a unit interval graph \(G\) of order \(n\), we will develop an \(O(n)\)-time algorithm for determining the existence of an unpaired \(k\)-DPC joining \(S\) and \(T\) and producing an unpaired \(k\)-DPC, if it exists, that provides a consecutive ordering of the vertices of \(G\) and a unit interval representation for \(G\) are both available. We then provide a reduction of the \textsc{General-Demand \(k\)-DPC} \([24]\) problem into the \textsc{Unpaired \(k\)-DPC} problem in \(O(n + k)\) time, so that the \textsc{One-to-One DPC} and the \textsc{One-to-Many DPC} problems are solvable both in \(O(n)\) time, provided that the two structures required by the unpaired \(DPC\) algorithm are available. Finally, we present an algorithm for the \textsc{Paired \(k\)-DPC} problem on a unit interval graph, which runs in time polynomial in \(n\) for a fixed \(k\) (where \(k\) is regarded as a constant).

2. Preliminaries

We begin with a consecutive ordering of the vertices of a unit interval graph \(G\), from which many interesting properties have been deduced. Hereafter, we denote by \(n\) the order of \(G\), i.e., \(n = |V(G)|\).

\textbf{Theorem 1 (Roberts \([34],[35]\)).} For a simple graph \(G\), the following statements are equivalent:

(a) \(G\) is a unit interval graph.

(b) \(G\) is a proper interval graph.

(c) There is a consecutive ordering, \((v_1, \ldots, v_n)\), of the vertices of \(G\).

\textbf{Theorem 2 (Looges and Olariu \([26]\)).} A simple graph \(G\) of order \(n\) is a unit interval graph if and only if there is an ordering, \((v_1, \ldots, v_n)\), of the vertices of \(G\), such that for all \(p < q < r\), \((v_p, v_q) \in E(G)\) implies \((v_p, v_q), (v_q, v_r) \in E(G)\).

\textbf{Lemma 1.} (See \([9],[11],[77]\), etc.) For an ordering \((v_1, \ldots, v_n)\) of the vertices of a unit interval graph \(G\), the followings are equivalent:

(a) The ordering is consecutive, i.e., the vertices of a maximal clique are consecutive.

(b) For all \(p < q < r\), \((v_p, v_q) \in E(G)\) implies \((v_p, v_q), (v_q, v_r) \in E(G)\).

(c) The vertices in the closed neighbor \(N_G(v_i)\) of a vertex \(v_i\) are consecutive.

Here, the \textit{closed neighborhood} \(N_G(v_i)\) of a vertex \(v_i\) refers to the set of vertices adjacent to \(v_i\) along with \(v_i\) itself, i.e., \(N_G(v_i) = \{v_i\} \cup \{v_j \in V(G) : (v_i, v_j) \in E(G)\}\). A graph is \(k\)-\textit{connected} if there is no set of fewer than \(k\) vertices whose removal results in a trivial graph or a disconnected graph. A \textit{hamiltonian path} is a path that passes through every vertex exactly once, and a \textit{hamiltonian cycle} is a cycle that passes through every vertex exactly once.
Theorem 3 (Chen, Chang, and Chang [9]). For any positive integer \( k \) and any proper interval graph \( G \) of \( n \geq k + 1 \) vertices with a consecutive ordering \((v_1, \ldots, v_n)\), \( G \) is \( k \)-connected if and only if \((v_i, v_j) \in E(G)\) whenever \( 1 \leq |i - j| \leq k \).

Theorem 4 (Bertossi [5]). A unit interval graph \( G \) of two or more vertices has a hamiltonian path if and only if \( G \) is connected.

Remark 1. If \((v_1, \ldots, v_n)\) is a consecutive ordering of the vertices of a 2-connected unit interval graph \( G \) of order \( n \geq 3 \), then \((v_1, v_3, \ldots, v_{n-1}, v_n, v_{n-2}, \ldots, v_2)\) will be a hamiltonian cycle of \( G \) for even \( n \); \((v_1,v_3,\ldots,v_n,v_{n-1},v_{n-3},\ldots,v_2)\) will be a hamiltonian cycle of \( G \) for odd \( n \). The hamiltonian cycle passes through both edges \((v_1,v_2)\) and \((v_{n-1},v_n)\), which implies that graph \( G \) has a hamiltonian path joining \( v_{n-1} \) and \( v_n \) as well as a hamiltonian path joining \( v_1 \) and \( v_2 \).

All graph-theoretical terms not defined here can be found in [6]. A path in a graph is represented as a sequence of vertices, in which any two consecutive vertices are adjacent. An interval graph \( G \) of \( n \) vertices is a left valley, called a valley if all the left vertices are contained in a path \( P \) and all the remaining paths are valley-free.

3. Properties of Many-to-Many DPCs

Let \((v_1, \ldots, v_n)\) be a consecutive ordering of the vertices of a unit interval graph \( G \). Vertex \( v_i \) is said to be to the left of \( v_j \) (or \( v_j \) is to the right of \( v_i \)) if \( i < j \). Let \( V_{i,j} = \{v_i, \ldots, v_j\} \) if \( i \leq j \) and let \( V_{i,j} = \emptyset \) if \( i > j \), so that \( V_{1,n} = V(G) \). We are given a source set, \( S = \{s_1, \ldots, s_k\} \), and a sink set, \( T = \{t_1, \ldots, t_k\} \), in \( G \) such that \( S \cap T = \emptyset \). Let \( p = \min \{i : v_i \in S \cup T\} \) and \( q = \max \{j : v_j \in S \cup T\} \), i.e., \( v_p \) is the lefmost terminal and \( v_q \) is the rightmost terminal.

For a path \( P \) that joins a source in \( S \) and a sink in \( T \), a left valley is a maximal subpath of \( P \), whose vertices are all contained in \( V_{1,p-1} \). If we delete all vertices in \( V_{p,n} \) from \( P \), then there remain zero or more left valleys that are pairwise disjoint. (See Fig. 3(a).) For a \( v_a-v_b \) subpath that is a left valley, called a left valley, there are two distinct vertices, \( v_{a'}, v_{b'} \in V_{p,n} \), such that \((v_{a'},v_{b'}),(v_a,v_b) \in E(P)\). The vertex \( v_{a'} \) is called the mate of \( v_a \), and \( v_{b'} \) is called the mate of \( v_b \). If the \( v_a-v_b \) valley is a one-vertex path, i.e., \( v_a = v_b \), it does not matter which vertex is the mate of \( v_a \). It is possible that \( v_a \) and \( v_b \), respectively, are matched with \( v_{a'} \) and \( v_{b'} \). In the same way, a right valley is a maximal subpath of \( P \), whose vertices are all contained in \( V_{q+1,n} \). The mates, \( v_{a'}, v_{b'} \in V_{1,q} \), of the end-vertices of a right valley are defined similarly.

Lemma 2. If \( G \) has an unpaired (resp. paired) \( k \)-DPC joining \( S \) and \( T \), then \( G \) has an unpaired (resp. paired) \( k \)-DPC in which at most one path has a left valley, at most one path has a right valley and all the remaining paths are valley-free.

Proof. We claim that for every \( v_a-v_b \) valley of a path \( P \) in the \( k \)-DPC, the mate \( v_{a'} \) of \( v_a \) and the mate \( v_{b'} \) of \( v_b \) are adjacent. Suppose the valley is a left one first. From the fact that \((v_{a'},v_a),(v_b,v_{b'}) \in E(G)\) and \( a < p < a' \), we have \((v_{a'},v_{b'}) \in E(G)\) for every \( p \leq i < a' \) by Lemma 1.
Symmetrically, we can conclude \( (v_{b'}, v_j) \in E(G) \) for every \( p \leq j < b' \). It follows that \( (v_{a'}, v_{b'}) \in E(G) \) whether \( a' < b' \) or \( b' < a' \). Symmetrically, we can conclude \( (v_{b'}, v_{p'}) \in E(G) \) for a right \( v_a-v_b \) valley, proving the claim. This implies that if we remove the \( v_a-v_b \) valley from the path \( P \) and join \( v_{b'} \) and \( v_p \) via an edge, we have a new path that joins the same source and sink as \( P \).

Suppose there are two left valleys, \( v_a-v_b \) valley and \( v_c-v_d \) valley, of some path(s) in the \( k \)-DPC. We will show that the two left valleys can be merged into one, resulting in a new \( k \)-DPC having one less left valley. (See Fig. 2 for an illustrative example.) We assume w.l.o.g. that \( a \leq b \), \( c \leq d \), and \( a < c \). Let the mates of \( v_a \), \( v_b \), \( v_c \) and \( v_d \) be \( v_{a'}, v_{b'}, v_{c'} \) and \( v_{d'} \), respectively. Then, \( (v_a, v_c), (v_d, v_{d'}) \in E(G) \) because the subgraph induced by \( V_{a,c} \) is a clique by the property of a consecutive ordering. It suffices to (i) remove the \( v_c-v_d \) valley of a path and join \( v_{c'} \) and \( v_{d'} \) via an edge, (ii) insert edge \( (v_a, v_{c'}) \) to merge the two valleys into one, a \( v_a-v_b \) valley, and (iii) replace the edge \( (v_a, v_{a'}) \) of the path that contains the old \( v_a-v_b \) valley with edge \( (v_{a'}, v_{a'}) \). Repeating this process, we conclude that there exists a \( k \)-DPC having at most one left valley. Similarly, we can repeatedly merge two right valleys into one, resulting in a \( k \)-DPC having at most one right valley. This completes the proof.

From Lemma 2, the left valley of an unpaired (or paired) \( k \)-DPC, if it is unique, forms a hamiltonian path of the subgraph \( G[V_1, p-1] \) induced by \( V_{1, p-1} \). Also, the right valley, if any, forms a hamiltonian path in the induced subgraph \( G[V_q, 1, n] \).

**Lemma 3.** Let \( G \) have an unpaired (resp. paired) \( k \)-DPC joining \( S \) and \( T \) that has at most one left valley and at most one right valley. Then, the following two hold true:

(a) If \( p \geq 2 \), there exists an unpaired (resp. paired) \( k \)-DPC that has a left \( v_a-v_b \) valley, where \( a = p - 2 \) and \( b = p - 1 \) for \( p \geq 3 \), and \( a = b = 1 \) for \( p = 2 \). Moreover, the mate of \( v_{a} \) is to the left of the mate of \( v_{b} \).

(b) If \( q \leq n - 1 \), there exists an unpaired (resp. paired) \( k \)-DPC that has a right \( v_c-v_d \) valley, where \( c = q + 1 \) and \( d = q + 2 \) for \( q \leq n - 2 \), and \( c = d = n \) for \( q = n - 1 \). Moreover, the mate of \( v_c \) is to the left of the mate of \( v_{d} \).

**Proof.** If \( p \in \{2, 3\} \), we have only one choice for the left valley: the one-vertex path \( (v_1) \) for \( p = 2 \) and the \( v_1-v_2 \) path for \( p = 3 \). Suppose \( p \geq 4 \). The subgraph \( G[V_1, p-1] \) is hamiltonian because the
left valley, say \( v_i \rightarrow v_j \), valley, becomes a hamiltonian cycle of \( G[V_{i,p-1}] \) if \( v_i \) and \( v_j \) are joined by an edge \((v_i, v_j) \in E(G)\). It follows, from Theorem 5 and Remark 1, that there exists a hamiltonian \( V_{p-2} \rightarrow V_{p-1} \) path in \( G[V_{i,p-1}] \). In the path of the DPC that contains the left \( v_i \rightarrow v_j \), valley, it is possible to switch the left valley with the hamiltonian \( (v_{p-2}, v_{p-1}) \) path (i.e., a new \( V_{p-2} \rightarrow V_{p-1} \) valley) through \((v_{p-2}, v_{p-1}), (v_{p-1}, v_{p})\) or \((v_{p-2}, v_{p}), (v_{p-1}, v_{p})\) for the mates \( v_{p-1} \) and \( v_{p-1} \), respectively.

This is because (i) if \( i < j \) (where \( i \leq p - 2 \) and \( j \leq p - 1 \)), then \((v_{p-2}, v_{p}), (v_{p-1}, v_{p})\) \in E(G), and (ii) if \( j < i \) (where \( j \leq p - 2 \) and \( i \leq p - 1 \)), then \((v_{p-2}, v_{p}), (v_{p-1}, v_{p})\) \in E(G). Finally, suppose the mate of \( v_a \) is to the right of the mate of \( v_b \), where \( a < b \). It suffices to swap their mates. The proof of (a) is complete. The proof of statement (b) is done similarly.

Remark 2. The left valley of Lemma 3(a) for \( p \geq 4 \) can be constructed easily from Remark 1 \((v_{p-1}, v_{p-3}, \ldots, v_2, v_1, v_3, \ldots, v_{p-2})\) for odd \( p \) and \((v_{p-1}, v_{p-3}, \ldots, v_1, v_2, v_4, \ldots, v_{p-2})\) for even \( p \). The same follows for the right valley of Lemma 3(b) for \( q \leq n - 3 \).

It follows immediately from Lemma 3 and Theorem 5 that:

Lemma 4. (a) For \( p \geq 4 \), \( G \) has an unpaired (resp. paired) \( k \)-DPC if and only if \( G[V_{p-2,n}] \) has an unpaired (resp. paired) \( k \)-DPC and \( G[V_{1,p-1}] \) is 2-connected.

(b) For \( q \leq n - 3 \), \( G \) has an unpaired (resp. paired) \( k \)-DPC if and only if \( G[V_{1,q+2}] \) has an unpaired (resp. paired) \( k \)-DPC and \( G[V_{q+1,n}] \) is 2-connected.

From the discussions of valleys in Lemmas 2 and 3 and from the property of a consecutive ordering, we also have that:

Lemma 5. (a) For every \( 1 \leq i < p \), the subgraph \( G[V_{i,n}] \) has an unpaired (resp. paired) \( k \)-DPC only if \( G[V_{i+1,n}] \) has an unpaired (resp. paired) \( k \)-DPC.

(b) For every \( q < j \leq n \), the subgraph \( G[V_{1,j}] \) has an unpaired (resp. paired) \( k \)-DPC only if \( G[V_{1,j-1}] \) has an unpaired (resp. paired) \( k \)-DPC.

4. Algorithm for the Unpaired DPC Problem

If we restrict our attention to the unpaired many-to-many DPCs of a unit interval graph \( G \), we can discover an additional property on valleys. The left valley, if any, can be associated with the leftmost terminal \( v_p \), and the right valley, if any, can also be associated with the rightmost terminal \( v_a \), as shown below.

Lemma 6. Let the graph \( G \) have an unpaired \( k \)-DPC joining \( S \) and \( T \), whose valleys are in shape of Lemma 3 Then, there exists an unpaired \( k \)-DPC such that

- the mate of \( v_a \) is \( v_p \) if there is a left \( v_a \rightarrow v_b \) valley with \( a \leq b \), and
- the mate of \( v_b \) is \( v_q \) if there is a right \( v_a \rightarrow v_d \) valley with \( c \leq d \).

Proof: Suppose that a path \( P \) in the \( k \)-DPC contains a left \( v_a \rightarrow v_b \) valley with \( a \leq b \), such that the mate of \( v_a \) is not \( v_p \). The mate of \( v_p \) is \( v_p \), neither. Let \( v_a \) and \( v_b \), respectively, be the mates of \( v_a \) and \( v_b \), where \( p < a' < b' \) by Lemma 3. Then, path \( P \) can be decomposed into three subpaths: the \( x \rightarrow v_a \) subpath, the \( v_a \rightarrow v_b \) valley and the \( v_b \rightarrow y \) subpath for some \( x, y \in S \cup T \). Notice that \((v_p, v_a), (v_p, v_b) \in E(G)\) and \((v_b, v_{a'}), (v_b, v_{b'}) \in E(G)\). There are two cases depending on whether \( v_p \in \{x, y\} \) or not.
Suppose \( v_p \notin \{x, y\} \) first. There exists a path \( P' \) (other than \( P \)) in the \( k \)-DPC, one of whose end-vertices is \( v_p \). Let \( v_p \) be the vertex next to \( v_p \) on \( P' \), i.e., \( P' = (v_p, v_p', P) \) for some subpath \( P \), possibly empty. We decompose path \( P' \) into two subpaths: one-vertex path \((v_p, P)\) and \((v_p', P)\). The five subpaths of \( P \) and \( P' \), in total, will be merged into two paths so as to obtain a new unpaired \( k \)-DPC. We have \( a', b', p' > p \) and moreover, the subgraph induced by \( \{v_\alpha, v_p, v_p'\} \) forms a clique of three vertices by Lemma 4. For the simplicity of our discussion, we assume \( v_p, x \in S \) and \( y \in T \). (The other cases can also be dealt with in the same manner.) If we concatenate the \( x-v_\alpha \) subpath and \((v_p, P)\) into one, and then concatenate \((v_p), \) the \( v_p-y \) valley and the \( v_p-y \) subpath into another, then we can obtain two paths, each of which runs from a source to a sink. If we let the other \( k-2 \) paths in the \( k \)-DPC remain unchanged, we have transformed the original \( k \)-DPC successfully into a new unpaired \( k \)-DPC that satisfies the first condition of the lemma.

Suppose \( v_p \in \{x, y\} \) now. Let \( v_p \) be the vertex next to \( v_p \) on \( P \). As before, we have \( a', b', p' > p \) and moreover, the subgraph induced by \( \{v_\alpha, v_p, v_p'\} \) forms a clique. Note that the induced graph does not necessarily have three vertices because possibly, \( p' = a' \) or \( p' = b' \). Assume \( x = v_p \). (The other case \( y = v_p \) can proved in the same way.) Then, \( p' \neq b' \) and \((v_p, v_p') \in E(G)\). We decompose the \( x-v_\alpha \) subpath once more into two subpaths: one-vertex path \((v_p)\) and the \( v_p-v_\alpha \) subpath. If we concatenate the four subpaths \((v_p), \) the \( v_\alpha-v_p \) valley, the \( v_\alpha-v_\alpha \) subpath and \( v_p-y \) subpath in sequence, then we have another \( x-y \) path. If we let the other \( k-1 \) paths in the \( k \)-DPC remain unchanged, we obtain a new unpaired \( k \)-DPC that satisfies the first condition of the lemma. In a symmetric way, we can transform the \( k \)-DPC that satisfies the first condition into a desired \( k \)-DPC, which satisfies the second condition as well as the first condition of the lemma. This completes the proof.

From the discussions of valleys of an unpaired DPC so far, our problem is reduced to the construction of an unpaired DPC in a unit interval graph whose leftmost and rightmost vertices are both terminals, as summarized below. Note that the class of unit interval graphs is hereditary, i.e., every induced subgraph of a graph in the class is contained in the same class.

**Lemma 7.** Let \( p \geq 2 \) or \( q \leq n-1 \). Then, graph \( G \) has an unpaired \( k \)-DPC joining \( S \) and \( T \) if and only if

- if \( p \geq 2 \), there exists a hamiltonian \( v_p-v_{p-1} \) path in the subgraph \( G[V_1, q]\);
- if \( q \leq n-1 \), there exists a hamiltonian \( v_q-v_{q+1} \) path in the subgraph \( G[V_q, n]\);
- the induced subgraph \( G' \) has an unpaired \( k \)-DPC joining \( S' \) and \( T' \), where

\[
G' = \begin{cases} 
G[V_{p-1} \cup v_{p+1} \cup \{v_{q+1}\}] & \text{if } p \geq 2 \text{ and } q \leq n-1, \\
G[V_{p-1} \cup \{v_{q+1}\}] & \text{if } p \geq 2 \text{ and } q = n, \\
G[V_{q+1} \cup \{v_{p+1}\}] & \text{if } p = 1 \text{ and } q \leq n-1;
\end{cases}
\]

\[
S' = \begin{cases} 
(S \setminus \{v_p, v_q\}) \cup \{v_p+1, v_q+1\} & \text{if } (p \geq 2 \land v_p \in S) \text{ and } (q \leq n-1 \land v_q \in S), \\
(S \setminus v_p) \cup \{v_{p-1}\} & \text{if } (p \geq 2 \land v_p \in S) \text{ and } (q \leq n-1 \land v_q \in S), \\
(S \setminus v_q) \cup \{v_{q+1}\} & \text{if } (p \geq 2 \land v_p \in S) \text{ and } (q \leq n-1 \land v_q \in S), \\
S & \text{if } (p \geq 2 \land v_p \in S) \text{ and } (q \leq n-1 \land v_q \in S);
\end{cases}
\]

\[
T' = \begin{cases} 
(T \setminus \{v_p, v_q\}) \cup \{v_{p-1}, v_{q+1}\} & \text{if } (p \geq 2 \land v_p \in T) \text{ and } (q \leq n-1 \land v_q \in T), \\
(T \setminus v_p) \cup \{v_{p-1}\} & \text{if } (p \geq 2 \land v_p \in T) \text{ and } (q \leq n-1 \land v_q \in T), \\
(T \setminus v_q) \cup \{v_{q+1}\} & \text{if } (p \geq 2 \land v_p \in T) \text{ and } (q \leq n-1 \land v_q \in T), \\
T & \text{if } (p \geq 2 \land v_p \in T) \text{ and } (q \leq n-1 \land v_q \in T).
\end{cases}
\]

**Remark 3.** For \( p \geq 2 \), the subgraph \( G[V_{1, p}] \) has a hamiltonian \( v_p-v_{p-1} \) path if and only if
For $q \leq n - 1$, the subgraph $G(V_{q,n})$ has a hamiltonian $v_q - v_{q+1}$ path if and only if
- if $q = n - 1$, the subgraph $G(V_{q,n})$ is connected; and
- if $q \leq n - 2$, the subgraph $G(V_{q,n})$ is 2-connected.

Now, we consider the unpaired $k$-DPC problem in a unit interval graph whose leftmost and rightmost vertices are both terminals, i.e., $v_1, v_n \in S \cup T$. One might expect at a glance that there always exists an unpaired $k$-DPC in which every path is monotone, where a path $(v_{i_1}, \ldots, v_{i_k})$ is said to be monotone if $i_1 < \cdots < i_l$ or $i_1 > \cdots > i_l$. A moment’s reflection will convince us that it cannot be so, as shown in Fig. 3. Let us define $h_m, 0 \leq m \leq n$, as the number of sources in $V_{1,m}$ minus the number of sinks in $V_{1,m}$, i.e.,

$$h_m := |S \cap V_{1,m}| - |T \cap V_{1,m}|,$$

where $h_0 = h_n = 0$. Similar to Fig. 3, we can construct an instance that admits an unpaired DPC but does not admit an unpaired DPC whose paths are all monotone, if there exists $j < j'$ such that $h_i = 0$ for all $j \leq i \leq j'$.

There are some instances that allow unpaired DPCs whose paths are all monotone. One of them will be the case when $h_m \geq 1$ for all $1 \leq m < n$, which will be clear in Lemma 8. We will devise an algorithm for the special case when $h_m \geq 1$ for all $1 \leq m < n$ ($v_1 \in S$ and $v_n \in T$), and then conquer our Unpaired DPC problem by dividing the input graph into subgraphs of the special case or their slight extensions. Let $X_m \subseteq V_{1,m}$ be the set of $h_m$ non-sink vertices (sources or non-terminals) whose indices are as large as possible. If we pick up $h_m$ non-sink vertices from $v_m$ to $v_1$ in a decreasing order of their indices, then we have the very set $X_m$. Let $X_m = \{x_1, \ldots, x_{h_m}\}$ where $x_i = v_q$, for $1 \leq i \leq h_m$ such that $a_1 < \cdots < a_{h_m}$. Similarly, let $Y_{m+1} \subseteq V_{m+1,n}$ be the set of $h_m$ non-source vertices (sinks or non-terminals) whose indices are as small as possible. Let $Y_{m+1} = \{y_1, \ldots, y_{h_m}\}$ where $y_i = v_h$, for $1 \leq i \leq h_m$ such that $b_1 < \cdots < b_{h_m}$.

**Lemma 8.** Let $h_m \geq 1$ for all $1 \leq m < n$. Then, graph $G$ has an unpaired $k$-DPC joining $S$ and $T$ if and only if $Z_m \subseteq E(G)$ for every $1 \leq m < n$, where

$$Z_m := \{(x_1, y_1), \ldots, (x_{h_m}, y_{h_m})\}.$$
If there are edges \((z, w)\) and \((z', w')\) in \(Z\), such that \(z\) is to the left of \(z'\) and \(w\) is to the right of \(w'\), then we replace the two edges with \((z, w')\) and \((z', w)\) in \(E(G)\). By repeating this process until no such pair exists, we get an edge set \(\{(z'_1, w'_1), \ldots, (z'_h, w'_h)\}\), where \(z'_1\) is to the left of \(z'_2\), \(z'_2\) is to the left of \(z'_3\), and so on, and \(w'_1\) is to the left of \(w'_2\), \(w'_2\) is to the left of \(w'_3\), and so on. Since \((z'_i, w'_i)\) \(\in E(G)\) implies \((x_i, y_i) \in E(G)\) for each \(i\), we have \(\{(x_1, y_1), \ldots, (x_{h+1}, y_{h+1})\} \subseteq E(G)\). Note that \(z'_1 = x_i\) or \(z'_2 = y_i\) or \(w'_1\) is to the right of \(y_i\).

Sufficiency will be proved by induction on \(n\). For the base case of \(n = 2\) (where \(v_1 = s_1\) and \(v_2 = t_1\)), we have \(Z_1 = \{(v_1, v_2)\} \subseteq E(G)\), which can be considered as a desired DPC. Suppose \(n \geq 3\). Let us denote by \(v_T\), the leftmost non-source vertex of \(G\), i.e., \(v_T \not\in S\) and \(\{v_1, \ldots, v_{r-1}\} \subseteq S\). There are two cases depending on whether \(v_T\) is a non-terminal or a sink.

Case 1: \(v_T\) is a non-terminal, i.e., \(v_T \not\in S \cup T\). We claim that the induced subgraph \(G[V(G) \setminus v_1]\) has an unpaired \(k\)-DPC joining \(S'\) and \(T\), where \(S' = (S \setminus v_T) \cup \{v_T\}\) and the vertex \(v_T\) is now a virtual source. Similarly, as we define \(h_m\) and \(Z_m\) for \(G\), we can define \(h'_m\) and \(Z'_m\) for the induced subgraph where \(1 \leq m < n\). (That is, \(h'_m = |S' \cap V_{2m}| = |T \cap V_{2m}|\), and \(Z'_m\) is a set of \(h'_m\) edges joining \(h'_m\) non-sink vertices in \(V_{2m}\) whose indices are as large as possible and \(h'_m\) non-source vertices in \(V_{m+1,n}\), whose indices are as small as possible.) Then, we have

\[
    h'_m = \begin{cases} 
    h_m - 1 & \text{if } 1 \leq m < l', \\
    h_m & \text{if } l' \leq m \leq n;
    \end{cases} \quad \text{and} \quad Z'_m = \begin{cases} 
    Z_m \setminus \{v_1, v_T\} & \text{if } 2 \leq m < l', \\
    Z_m & \text{if } l' \leq m < n.
    \end{cases}
\]

Furthermore, \(h'_m \geq 1\) for all \(2 \leq m < n\). Thus, there exists an unpaired \(k\)-DPC joining \(S'\) and \(T\) in the induced subgraph \(G[V(G) \setminus v_1]\) by the induction hypothesis, thereby proving the claim.

Case 2: \(v_T\) is a sink, i.e., \(v_T \in T\). Since \(h_T \geq 1\), the vertex \(v_T\) in this case should be a source, so that \(k \geq 2\) and \(l' \geq 3\). Consider the induced subgraph \(G[V(G) \setminus \{v_1, v_T\}]\). Similar to Case 1, we can see, by the induction hypothesis, that the induced subgraph has an unpaired \((k-1)\)-DPC joining \(S'\) and \(T'\), where \(S' = S \setminus v_1\) and \(T' = T \setminus v_T\). Note that \(h'_m\) and \(Z'_m\) for this induced subgraph are as follows:

\[
    h'_m = \begin{cases} 
    h_m - 1 & \text{if } 1 \leq m < l', \\
    h_m & \text{if } l' \leq m \leq n;
    \end{cases} \quad \text{and} \quad Z'_m = \begin{cases} 
    Z_m \setminus \{v_1, v_T\} & \text{if } 2 \leq m < l', \\
    Z_m & \text{if } l' \leq m < n.
    \end{cases}
\]

Adding the path \((v_1, v_T)\) to the \((k-1)\)-DPC results in a desired \(k\)-DPC of \(G\) joining \(S\) and \(T\). This completes the proof.

According to the proof of Lemma\[9\] we can design a simple \(O(n)\)-time algorithm for finding an unpaired DPC for the special case where \(h_m \geq 1\) for all \(1 \leq m < n\). It is worth noting that at all times, from the current leftmost vertex, which is a (virtual) source, we advance to the leftmost non-source vertex. This implies that every path in the DPC is monotone. By virtue of Lemma\[7\] moreover, the algorithm can be adapted to a slightly more general case where \(p\) is not necessarily 1 and \(q\) is not necessarily \(n\), i.e., the case when \(h_m \geq 1\) for every \(p \leq m < q\) (and \(h_m = 0\) for \(m < p\) or \(m \geq q\)).

The algorithm can be summarized as a procedure below, named \(\text{Find-UDPC}\), that computes an unpaired DPC joining \(S'\) and \(T'\) in the induced subgraph \(G[V_{2,m}]\), in which \(v_p\) and \(v_q\), respectively, are the leftmost and rightmost terminals. The procedure is illustrated in Fig.\[4\]. We assume that array \(L[1..n]\) is precomputed, where \(L[i] = \text{nonterminal, source and sink}\), respectively, if \(v_i\) is a non-terminal, source and sink.

9
Fig. 4: The unpaired 2-DPC and 1-DPC produced by procedure Fino-UDPC-A.

1: procedure Fino-UDPC-A($S'$, $T'$, $l$, $r$, $p$, $q$, $I$)  
2:   if $p = l$ then  
3:     Initialize the $v_p$-path to be a one-vertex path ($v_p$);  
4:     $i \leftarrow p$;  
5:   else  
6:     Initialize the $v_p$-path to be the extended valley returned by Fino-Left-Valley($l$, $p$);  
7:     $i \leftarrow p - 1$;  
8:     $L[p] \leftarrow$ discarded;  
9: end if  
10: for all $s \in S'$ other than $v_p$ do  
11:     Initialize the $s$-path to be a one-vertex path ($s$);  
12: end for  
13: $l' \leftarrow i$;  
14: repeat  
15:     Find the leftmost non-source vertex, $v_l'$, to the right of $v_i$.  
16:     $l' \leftarrow l' + 1$;  
17: until $L[l'] \in \{\text{nonterminal, sink}\}$  
18: while $i < q$ do  
19:     Append $v_l'$ to the path ending at $v_l'$;  
20:     if $L[l'] = \text{nonterminal}$ then  
21:         $L[l'] \leftarrow$ source;  
22:     else  
23:         $L[l'] \leftarrow$ discarded;  
24:     end if  
25:     Find the next (virtual) source, $v_i$.  
26:     $i \leftarrow i + 1$;  
27: until $L[i] = \text{source} \lor i \geq q$  
28: repeat  
29:     Find the leftmost non-source vertex, $v_l'$, to the right of $v_i$.  
30:     $l' \leftarrow l' + 1$;  
31: until $L[l'] \in \{\text{nonterminal, sink}\} \lor l' \geq q$

\(\triangleright\) Assume $h_{m} \geq 1 \forall p \leq m \leq q$. 
\(\triangleright\) There exists no left valley. 
\(\triangleright\) There exists a left valley. 
\(\triangleright\) $v_i$ is the leftmost (virtual) source. 
\(\triangleright\) Hereafter, $v_p$ will be ignored.
The existence of an unpaired $k'$ subgraph has an unpaired $k$ next (virtual) sources $v$ procedures run in time linear to $\mid k'\mid$.

Lemma 9. For the case where $h_m \geq 1$ for every $p \leq m < q$, procedure FIND-UDPC-A determines the existence of an unpaired $k'$-DPC joining $S'$ and $T'$ in the induced subgraph $G[V_{l_r}]$ and produces a $k'$-DPC, if it exists, in time linear to the number of vertices of the subgraph, where $k' = \mid S'\mid$.

Proof. From the discussions in Lemmas 7 and 8 and in Remarks 2 and 3, there always exists an unpaired $k'$-DPC that is exactly the same as that which the procedure produces if the induced subgraph has an unpaired $k'$-DPC. It follows that the procedure is correct. The valley finding procedures run in time linear to $\mid V_{l_{p+1}}\mid + \mid V_{q+1}\mid$. Moreover, the total time for finding all the next (virtual) sources $v_i$ (in lines 4, 7, and 24–26) and all the next $v_r$ (in lines 13–16 and 27–29) is linear to $\mid V_{p,q}\mid + 1$. Finally, the query if $(v_f, v_f) \in E(G)$ can be answered in $O(1)$ time from the interval representation $I$, so the validity check (in lines 35–39) can be done in linear-time to $\mid V_{l_r}\mid$. Therefore, the running time of the procedure is linear to $\mid V_{l_r}\mid$, i.e., $O(\mid V_{l_r}\mid)$.

Remark 4. Let $v_r$ be the rightmost non-sink vertex of $V_{p,q}$ if $l = p$, or of $\{V_{p-1}\} \cup V_{p+1,q}$ if $l < p$.

Procedure FIND-UDPC-A produces an unpaired $k'$-DPC, whose right valley, if any, is, associated with $v_r$ as well as $v_q$, i.e., $v_r$ and $v_q$ are the two mates of end-vertices of the right valley.

Remark 5. Symmetric to procedure FIND-UDPC-A, we can also develop a procedure, named FIND-UDPC-B, applicable to the case where $h_m \leq -1$ for every $p \leq m < q$. (It suffices to switch the roles of sources and sinks.)
There remains a case where \( h_m = 0 \) for some \( p \leq m < q \). (Note that if \( h_{m'} > 0 \) and \( h_{m'} < 0 \) for some \( p \leq m', m'' < q \), then there exists such \( m \) in the range.) To deal with the general cases of our \textsc{Unpaired} DPC problem, one of the natural approaches we can think of would be to decompose graph \( G \) into a number of induced subgraphs to each of which one of the procedures \textsc{Find-UDPC-A} or \textsc{Find-UDPC-B} is applicable. The lemma below suggests that this approach is plausible.

**Lemma 10.** Let \( q' \) and \( p' \) be the indices such that \( v_{q'p'}, v_{p'p} \in S \cup T \), \( q' \leq p' \) and \( h_m = 0 \) for all \( q' \leq m < p' \). Suppose the pair, \( q' \) and \( p' \), is the leftmost one, i.e., \( h_i \neq 0 \) for all \( p \leq i < q' \). Then, graph \( G \) has an unpaired \( k \)-DPC joining \( S \) and \( T \) if and only if for some \( q \leq m < p \), the subgraph, \( G_1 \), induced by \( V_{1,m} \) has an unpaired \( k_1 \)-DPC joining \( S_1 \) and \( T_1 \) and moreover, the subgraph, \( G_2 \), induced by \( V_{m+1,q} \) has an unpaired \( k_2 \)-DPC joining \( S_2 \) and \( T_2 \), where \( S_1 = S \cap V(G_i) \), \( T_1 = T \cap V(G_i) \), and \( k_i = |S_i| \) for \( i \in \{1, 2\} \).

**Proof.** The sufficiency proof is straightforward because the union of an unpaired \( k_1 \)-DPC of \( G_1 \) and an unpaired \( k_2 \)-DPC of \( G_2 \) becomes an unpaired \( k \)-DPC of \( G \). To prove necessity, suppose graph \( G \) has an unpaired \( k \)-DPC joining \( S \) and \( T \), and let \( m \) be \( q' \leq m < p' \). If there is a path, \( P_1 \), in the DPC joining a source \( s \in V(G_1) \) and a sink \( t \in V(G_2) \), then there is also a path, \( P_2 \), joining a source \( s' \in V(G_2) \) and a sink \( t' \in V(G_1) \). Let \( (x, y) \in E(P_1) \), where \( x \in V(G_1) \) and \( y \in V(G_2) \), be the first edge encountered when we traverse \( P_1 \) starting at \( s \). Also, let \( (x', y') \in E(P_2) \), where \( x' \in V(G_2) \) and \( y' \in V(G_1) \), be the first edge encountered when we traverse \( P_2 \) starting at \( s' \). Then, we have \( (x, y), (x', y') \in E(G) \) by Lemma\footnote{Replacing \( [(x, y), (x', y')] \) with \( [(x, y'), (x', y)] \) in \( P_1 \) and \( P_2 \) results in the \( s-t' \) path and \( s'-t \) path, which reduces the number of paths in the DPC that join a pair of source and sink, one in \( G_1 \) and the other in \( G_2 \). If we repeat this process, we can obtain an unpaired DPC in which every path joins a source and a sink, both in \( G_1 \) or both in \( G_2 \).}.

The DPC may still have a path that joins a source and a sink both contained in \( G_2 \) but passes through some vertices in \( G_1 \). For a path \( P \) whose source and sink are both in \( G_2 \), we refer to a maximal subpath of \( P \) whose vertices are all contained in \( G_1 \) as a \textit{left valley}, within the scope of this proof. Similarly, the term \textit{right valley} will be used for path \( P' \) whose source and sink are both in \( G_1 \). In the same way as the proof of Lemma\footnote{Note that if \( h_s = 0 \) is the leftmost one, then \( (x, y) \in E(G) \) by Lemma\footnote{that transformation does not alter the source-sink matching, i.e., if a source \( j_i \) is matched with a sink \( j_i \) in the old DPC, then \( j_i \) is still matched with \( j_i \) in the new DPC.} that the transformation does not alter the source-sink matching, i.e., if a source \( j_i \) is matched with a sink \( j_i \) in the old DPC, then \( j_i \) is still matched with \( j_i \) in the new DPC.} we can transform the DPC into a new DPC with at most one left valley and at most one right valley in total. (We note, for reuse of this lemma for the \textsc{Paired} \( k \)-DPC problem in Section\footnote{that the transformation does not alter the source-sink matching, i.e., if a source \( j_i \) is matched with a sink \( j_i \) in the old DPC, then \( j_i \) is still matched with \( j_i \) in the new DPC.} that the transformation does not alter the source-sink matching, i.e., if a source \( j_i \) is matched with a sink \( j_i \) in the old DPC, then \( j_i \) is still matched with \( j_i \) in the new DPC.) Suppose there are two valleys: one left valley, say \( v_{a-b} \), with \( a \leq b \), and one right valley, say \( v_{c-d} \), with \( c \leq d \). For the mates \( v_{a'v_{a+b}} \) and \( v_{b'v_{b}} \), respectively, the subgraph induced by \( \{v_{a'}, v_{b'}, v_{a}, v_{b}\} \) forms a clique because \( (v_{a'}, v_{a+b}) \), \( (v_{b'}, v_{a+b}) \), \( (v_{a}, v_{b}) \), \( (v_{b'}, v_{b}) \), \( (v_{a'}, v_{a}) \), \( (v_{b'}, v_{b'}) \), \( (v_{a}, v_{b}) \), \( (v_{b'}, v_{b}) \), \( (v_{a'}, v_{a'}) \), \( (v_{b'}, v_{b'}) \), \( (v_{a}, v_{b}) \), \( (v_{b'}, v_{b}) \). Note that \( \{v_{a'}, v_{b'}, v_{a+b}, v_{a}, v_{b}, v_{b}, v_{a'}, v_{b'}, v_{a}, v_{b}, v_{a'}, v_{b'}\} = 0 \) because no path contains the left and right valleys simultaneously. Then, the new DPC is valley-free.

Finally, suppose there exists only one valley, say left valley, in the DPC. Let \( m' \) be the smallest index such that \( V(P) \subseteq V_{1,m'} \) for every path \( P \) that joins a source and a sink in \( G_1 \). Obviously, we have \( q' \leq m' \leq m \) (because we have no right valley). We will transform the DPC into a new one, which is the same as the union of an unpaired \( k_1 \)-DPC of \( G'_1 \), the induced subgraph \( G[V_{1,m'}] \), and an unpaired \( k_2 \)-DPC of \( G'_2 \), the induced subgraph \( G[V_{m'+1,q}] \). If there exists no left valley with
respect to $G'_1$ and $G'_2$, then we are done. Suppose there exists a left valley, say $v_a$-$v_b$, valley, in the DPC. For the mates $v_{a'}$ and $v_{b'}$ of $v_a$ and $v_b$, respectively, if we remove the valley from a path that contains it and then join $v_{a'}$ and $v_{b'}$ by an edge, there remains an isolated valley, the $v_a$-$v_b$ valley. Let $v_{a'} \in V(P)$ for some path $P$ in the DPC, and let $(v_{a'}, v_{b'}) \in E(P)$ for some vertex $v_{a'}$. Note that path $P$ joins a source and a sink in $G'_1$. Similar to the previous case where there are one left valley and one right valley, we can deduce that $G[[v_{a'}, v_{b'}, v_a, v_b]]$ forms a clique. If we delete $(v_{a'}, v_{b'})$ from $P$ and add $(v_{a'}, v_a)$ and $(v_b, v_{b'})$, then the new path $P$ passes through the valley. This completes the entire proof.

The above lemma indicates that, with respect to some $m$ with $q' \leq m < p'$, our DPC problem can be divided into two: one (special) DPC problem on $G[V_{1,m}]$ and another (possibly general) DPC problem on $G[V_{m+1,n}]$. We are to pick up a specific $m$ with respect to which our problem is divided into two. Lemmas 7 and 8 suggest that if $m \geq q' + 1$, it is necessary for $G_1$ to have an unpaired $k_1$-DPC that $(v_{q'+1}, v_r)$ is an edge of $G$, where $v_r$ is the vertex defined in Remark 4 when $v_r$ is a sink, i.e., (i) for the case of $v_r \in T$, the vertex $v_r$ is the rightmost non-sink vertex of $V_{p,q'}$ if $p = l$, or of $[v_{p-1}] \cup V_{p+1,q'}$ if $p > l$; (ii) for the case of $v_r \in S$, the vertex $v_r$ is the rightmost non-source vertex of $V_{p,q'}$ if $p = l$, or of $[v_{p-1}] \cup V_{p+1,q'}$ if $p > l$. Define

$$m^* = \begin{cases} q' & \text{if } q' + 1 = p' \text{ or } (v_{q'+1}, v_r) \not\in E(G), \\ q'' & \text{otherwise,} \end{cases}$$

where $q''$ is the maximum over all $j, q' + 1 \leq j < p'$, subject to that $G[V_{q',j}]$ has a hamiltonian $v_q - v_{q'+1}$ path.

**Lemma 11.** Let $q'$ and $p'$ be the pair of the leftmost indices of Lemma 10 so that $v_{q'}, v_{p'} \in S \cup T$, $q' < p'$, $h_n = 0$ for all $q' \leq m < p'$, and $h_i \neq 0$ for all $p \leq i < q'$. Then, graph $G$ has an unpaired $k$-DPC joining $S$ and $T$ if and only if the subgraph, $G'_1$, induced by $V_{1,m}$, has an unpaired $k_1$-DPC joining $S_1$ and $T_1$, and moreover, the subgraph, $G'_2$, induced by $V_{m+1,n}$, has an unpaired $k_2$-DPC joining $S_2$ and $T_2$, where $S_i = S \cap V(G'_i)$, $T_i = T \cap V(G'_i)$, and $k_i = |S_i|$ for $i \in \{1, 2\}$.

**Proof.** The sufficiency part of the proof is obvious. To show the necessity part, by Lemma 10 we assume that for some $m$, where $q' \leq m < p'$, the subgraph, $G_1$, induced by $V_{1,m}$, has an unpaired $k_1$-DPC joining $S_1$ and $T_1$, and moreover, the subgraph, $G_2$, induced by $V_{m+1,n}$, has an unpaired $k_2$-DPC joining $S_2$ and $T_2$. We claim $m \leq m^*$. Suppose $m > m^*$ for a contradiction. It follows that $m > q'$, and thus $(v_{q'+1}, v_r) \in E(G)$. Furthermore, by Lemma 7, $G[V_{q',m}]$ has a hamiltonian $v_q - v_{q'+1}$ path. This implies that $m \leq q' = m^*$, leading to a contradiction. Thus, the claim is proved. Suppose $m < m^*$. (We have nothing to prove if $m = m^*$.) It follows that $m^* > q'$. By the definition of $m^*$ in (1), we have $q' + 1 < p'$, $(v_{q'+1}, v_r) \not\in E(G)$, and $G[V_{q',m}]$ has a hamiltonian $v_q - v_{q'+1}$ path. Thus, $G'_1$ has an unpaired $k_1$-DPC joining $S_1$ and $T_1$ by Lemma 7. Also, $G'_2$ has an unpaired $k_2$-DPC joining $S_2$ and $T_2$ by Lemma 5(a). Therefore, the lemma is proved.

Now, we are ready to present an algorithm for finding an unpaired $k$-DPC of $G$. It is assumed that we are given a consecutive ordering $(v_1, \ldots, v_n)$ of the vertices of $G$ and a unit interval representation $I$ for $G$. Some running examples of the algorithm can be found in Fig. 5.

1: **procedure** FIND-UNPAIRED-DPC($S, T, (v_1, \ldots, v_n), I$)
2: \[ P \leftarrow \emptyset; \ h_0 \leftarrow 0; \] \hspace{1cm} » Initialize DPC $P$ to be empty.
Fig. 5: Running examples of procedure \textit{Find-Unpaired-DPC} for three configurations of sources and sinks on the graph shown in Fig. (a).

3: \( l \leftarrow 1; p \leftarrow 0; S' \leftarrow \emptyset; T' \leftarrow \emptyset; rn \leftarrow 0; m \leftarrow 1; \) \hspace{0.5cm} \( \triangleright \) Set \( p \) and \( rn \) to be undefined.
4: \textbf{while} \( m \leq n \) \textbf{do}
5: \hspace{0.5cm} \textbf{if} \( L[m] = \text{source} \) \textbf{then}
6: \hspace{1cm} \( h_m \leftarrow h_{m-1} + 1; S' \leftarrow S' \cup \{v_m\}; \)
7: \hspace{1cm} \( rs \leftarrow m; \) \hspace{0.5cm} \( \triangleright rs \leftarrow \) the index of the rightmost source so far
8: \hspace{1cm} \textbf{if} \( p = 0 \) \textbf{then} \( p \leftarrow m; \) \hspace{0.5cm} \( \triangleright p \leftarrow \) the index of the leftmost non-terminal
9: \hspace{1cm} \textbf{else if} \( L[m] = \text{sink} \) \textbf{then}
10: \hspace{1.5cm} \( h_m \leftarrow h_{m-1} - 1; T' \leftarrow T' \cup \{v_m\}; \)
11: \hspace{1.5cm} \( rt \leftarrow m; \) \hspace{0.5cm} \( \triangleright rt \leftarrow \) the index of the rightmost sink so far
12: \hspace{1.5cm} \textbf{if} \( p = 0 \) \textbf{then} \( p \leftarrow m; \) \hspace{0.5cm} \( \triangleright p \leftarrow \) the index of the leftmost non-terminal
13: \hspace{1.5cm} \textbf{else}
14: \hspace{2cm} \( h_m = h_{m-1}; \)
15: \hspace{2cm} \( rn \leftarrow m; \) \hspace{0.5cm} \( \triangleright rn \leftarrow \) the index of the rightmost non-terminal so far
16: \hspace{1cm} \textbf{end if}
17: \hspace{0.5cm} \textbf{if} \( h_m = 0 \) \textbf{and} \( L[m] = \text{sink} \) \textbf{then} \hspace{0.5cm} \( \triangleright \) Procedure \textit{Find-UDPC-A} is applicable.
18: \hspace{1cm} \( q' \leftarrow m; \) \hspace{0.5cm} \( \triangleright q' \leftarrow \) the index of the rightmost non-terminal
19: \hspace{1cm} \( r^* \leftarrow \max\{rs, rn\}; \) \hspace{0.5cm} \( \triangleright r^* \leftarrow \) the index of the rightmost non-sink so far
20: \hspace{1cm} \textbf{if} \( r^* = p \) \textbf{and} \( l < p \) \textbf{then} \( r^* \leftarrow p - 1; \) \hspace{0.5cm} \( \triangleright v_{r^*} \) is now the vertex of Remark 4
21: \hspace{1cm} \textbf{if} \( q' = n, L[q'+1] \in \{\text{source}, \text{sink}\}, \) \textbf{or} \( (v_{q'+1}, v_{r^*}) \notin E(G) \) \textbf{then} \hspace{0.5cm} \( \triangleright \) Determine \( m'. \)
22: \hspace{1cm} \( m^* \leftarrow q'; \)
23: \hspace{1cm} \textbf{else}
24: \hspace{1.5cm} \( m^* \leftarrow q' + 1; \)
while \( m^* + 1 \leq n, L[m^* + 1] = \text{nonterminal}, \) and \((v_{m^* + 1}, v_{m^* - 1}) \in E(G)\) do
\[
m^* \leftarrow m^* + 1
\]
end while

end if
\[
Pr' \leftarrow \text{FIND-UDPC-A}(S', T', l, m^*, p, q', T);
\]
if \( Pr' \) is a valid unpaired DPC then
\[
Pr \leftarrow Pr \cup Pr' ;
\]
else
return a message saying "there exists no unpaired DPC";
end if
\[
thm^* = 0;
\]
l \leftarrow \( m^* + 1 \); p \leftarrow 0; S' \leftarrow 0; T' \leftarrow 0; m \leftarrow m^* + 1;
else if \( h_{m} = 0 \) and \( L[m] = \text{source} \) then \( \triangleright \) Procedure FIND-UDPC-B is applicable.
Similar to lines 18 through 36 an unpaired DPC can be determined and found;
else \( \triangleright \) Here, \( h_{m} \neq 0 \) or \( L[m] = \text{nonterminal} \).
\[
m \leftarrow m + 1;
\]
end if
\[
\]
end while
\[
\]
if \( l = n + 1 \) then \( \triangleright \) All the vertices \( v_1 \) through \( v_n \) are processed.
\[
\]
return \( Pr \);
else
return a message saying “there exists no unpaired DPC”;
end if
\[
\]
end procedure

Theorem 6. Procedure FIND-UNPAIRED-DPC determines the existence of an unpaired DPC joining \( S \) and \( T \) in a unit interval graph \( G \) and computes the DPC, if any, in \( O(n) \) time when we are given a consecutive ordering of the vertices of \( G \) and a unit interval representation for \( G \).

Proof. The correctness of the procedure is due to Lemmas 9 and 11 and Remark 5. To extract a subproblem from the remaining DPC problem on \( G[V_{m^*}] \), we find \( v_p \) and \( v_{q'} \) (along with \( S' \) and \( T' \)) in time linear to \( |V_{m^*}| \) and then determine \( m^* \) (in lines 21–28) in time linear to \( |V_{q, m'}| \). Once a subproblem is identified in \( O(|V_{m^*}|) \) time, it is solved through one of the procedures FIND-UDPC-A and FIND-UDPC-B also in \( O(|V_{m^*}|) \) time, totally in time linear to the number of vertices of the subproblem. Thus, the procedure runs in \( O(n) \) time.

Remark 6. The condition \( h_{m} = 0 \) in lines 17 and 37 of procedure FIND-UNPAIRED-DPC may be replaced with \( |S'| = |T'| \). As a result, all the statements concerning with \( h_{m} \) may be removed.

The algorithm presented in this section is implemented. The source code and some running examples may be downloaded from http://tcs.catholic.ac.kr/~jhpark/papers/UDPC_in_UIG.zip

5. Reduction of GENERAL-DEMAND DPC Problem

The GENERAL-DEMAND DPC problem, suggested in [24], is a generalization of the one-to-one, one-to-many and unpaired many-to-many DPC problem. Given two pairwise disjoint terminal
sets, a source set \( S = \{s_1, \ldots, s_k\} \) and a sink set \( T = \{t_1, \ldots, t_m\} \), in which each source and sink is associated with the integer-valued demand \( d(t) \geq 1 \) such that \( \sum_{s \in S} d(s) = \sum_{t \in T} d(t) = k \), a general-demand \( k\)-DPC joining \( S \) and \( T \) is a \( k\)-DPC, in which \( d(s_i) \) paths have \( s_i \) as their common source for all \( s_i \in S \), and \( d(t_j) \) paths have \( t_j \) as their common sink for all \( t_j \in T \). The general-demand \( k\)-DPC becomes one-to-one, one-to-many and unpaired many-to-many, respectively, if \( k' = k'' = 1 \) (\( d(s_i) = d(t_j) = k \)), if \( k' = 1 \) and \( k'' = k \) (\( d(s_i) = k \) and \( d(t_j) = 1 \) for all \( t_j \in T \)), and if \( k' = k'' = k \) (\( d(s_i) = 1 \) for all \( s_i \in S \) and \( d(t_j) = 1 \) for all \( t_j \in T \)).

Consider a General-Demand \( k\)-DPC problem on a unit interval graph \( G \) with source set \( S \) and sink set \( T \), in which we are given a consecutive ordering \( (v_1, \ldots, v_n) \) of the vertices of \( G \) and a unit interval representation \( I \) for \( G \). Suppose there exists a terminal (source or sink) whose demand is two or greater. Then, the problem can be reduced to a general-demand \( k\)-DPC problem on another unit interval graph \( G' \), in which the number of terminals whose demand is at least two is one less than that of the original problem on \( G \), as follows: Let the demand of some terminal, say source \( s_i \), be at least two, and let \( s_i = v_r \) for some \( r \). If we replace the interval \( I_r \) for \( s_i \) with \( d(s_i) \) copies of \( I_r, I_{r+1}, \ldots, I_{r+d(s_i)-1} \) from \( I \), we obtain an interval representation \( I' \) for \( G' \). Let \( S' = (S \setminus s_i) \cup \{s_{i,1}, \ldots, s_{i,d(s_i)}\} \), where \( s_{i,j} \) is the vertex \( v_{r+j} \) that corresponds to interval \( I'_{r+j} \), and set \( d(s_{i,j}) = 1 \) for all \( 1 \leq j \leq d(s_i) \). Also, \( (v_1, \ldots, v_{r-1}, v_r, v_{r+1}, \ldots, v_{r+d(s_i)-1}, v_{r+d(s_i)}, v_{r+d(s_i)+1}, \ldots, v_n) \) will be a consecutive ordering of the vertices of \( G' \). See Fig. 6 for an illustrative example. It is straightforward to see that \( G' \) has a general-demand \( k\)-DPC joining \( S' \) and \( T \) if and only if \( G \) has a general-demand \( k\)-DPC joining \( S \) and \( T \). Furthermore, a general-demand \( k\)-DPC of \( G \) can be obtained from the general-demand \( k\)-DPC of \( G' \) by contracting the \( d(s_i) \) sources, \( s_{i,1}, \ldots, s_{i,d(s_i)} \), into a single source \( s_i \). Repeating this process eventually reduces the General-Demand \( k\)-DPC problem to the Unpaired \( k\)-DPC problem (without terminals of demand at least two). Therefore, we have:

**Theorem 7.** The General-Demand \( k\)-DPC problem on a unit interval graph \( G \) of order \( n \) can be solved in \( O(n+k) \) time if we are given a consecutive ordering of the vertices of \( G \) and a unit interval representation for \( G \).

**Proof.** The aforementioned reduction of the General-Demand \( k\)-DPC problem to the Unpaired \( k\)-DPC problem can be completed in \( O(n+k) \) time. Thus, the theorem follows from Theorem 6.

**Remark 7.** For graph \( G \) to have a general-demand \( k\)-DPC joining \( S \) and \( T \), it is necessary that \( k \leq n^2/4 \). If \( f(n) \) denotes the maximum \( k \), over all \( n \)-vertex graphs along with \( S \) and \( T \), such that
For the paired \( k \)-DPC problem on a unit interval graph \( G \), we assume w.l.o.g. that each source \( s_i \) is to the left of its designated sink \( t_i \) for \( i \in \{1, \ldots, k\} \). (Suppose otherwise, it suffices to exchange \( s_i \) and \( t_i \).) By virtue of Lemmas 2 and 3, we can restrict our attention to the paired DPCs having the unique left valley, if any, and the unique right valley, if any, in shape of Lemma 3. The approach taken for the unpaired DPC problem in Section 4 is, however, no longer applicable to the paired DPC problem due to the following: First, the left valley is not always associated with the leftmost terminal \( v_p \), which is in fact a source by our assumption, as shown in Fig. 7. Similarly, the right valley is not always associated with the rightmost terminal \( v_q \); Second, it is hard to devise a counterpart of Lemma 8 as suggested by the two instances shown in Fig. 8.

To solve our Paired \( k \)-DPC problem, we first divide the problem into smaller subproblems. Let \( q' \) and \( p' \) be the pair of the leftmost indices of Lemma 10 so that \( v_{q'}, v_{p'} \in S \cup T \), \( q' < p' \), \( h_m = 0 \) for all \( q' \leq m < p' \), and \( h_m \neq 0 \) for all \( p \leq m < q' \). Recall \( h_m \) is defined as ...
\( |S \cap V_{1,m}| - |T \cap V_{1,m}| \). By our assumption that every \( s_i \) is to the left of \( t_i \), we have \( h_m \geq 0 \) for all \( 0 \leq m \leq n \). Also, we have that for every \( i \), either \( s_i, t_i \in V_{p,q'} \) or \( s_i, t_i \notin V_{p,q'} \). It is fortunate for us to recycle Lemma 10 (although Lemma 11 is not reusable). More specifically, it holds true the counterpart of Lemma 10 for the Paired \( k \)-DPC problem asserting that graph \( G \) has a paired \( k \)-DPC joining \( S \) and \( T \) if and only if for some \( q' \leq m < p' \), the subgraph, \( G_1 \), induced by \( V_{1,m} \) has a paired \( k_1 \)-DPC joining \( S_1 \) and \( T_1 \) and moreover, the subgraph, \( G_2 \), induced by \( V_{m+1,n} \) has a paired \( k_2 \)-DPC joining \( S_2 \) and \( T_2 \) (where \( S_1 = S \cap V(G_i) \), \( T_1 = T \cap V(G_i) \), and \( k_i = |S_i| \) for \( i \in \{1, 2\} \)). Actually, the second and third paragraphs of the proof of Lemma 10 serve as a necessity proof of the counterpart (whereas the sufficiency is obvious). This is because it never occurs the case when there exists a path in the paired DPC of \( G \) that joins a source in \( V_{1,m} \) and a sink in \( V_{m+1,n} \) or joins a source in \( V_{m+1,n} \) and a sink in \( V_{1,m} \).

Instead of relying on Lemma 11 to pick up a specific \( m \) in the range \( q' \leq m < p' \) for dividing the problem into two parts, we let

\[
m'' = \begin{cases} 
q' + 1 & \text{if } q' + 1 = p' \text{ or } G[V_{1,q'+1}] \text{ has no paired DPC,} \\
q' + 2 & \text{if } q' + 2 = p' \text{ or } G[V_{1,q'+2}] \text{ has no paired DPC,} \\
q'' & \text{elsewhere,}
\end{cases}
\]

where \( q'' \) is the maximum over all \( j \), \( q' + 2 \leq j < p' \), subject to the fact that \( G[V_{q'+1,j}] \) has a hamiltonian \( v_{q'+1,v_{q'+2}} \) path. Refer to Fig. 9 for illustrative examples.

**Lemma 12.** Let \( q' \) and \( p' \) be the pair of the leftmost indices such that \( v_{q'}, v_{p'} \in S \cup T \), \( q' < p' \), \( h_m = 0 \) for all \( q' \leq m < p' \), and \( h_j \geq 1 \) for all \( p \leq j < q' \). Graph \( G \) has a paired \( k \)-DPC joining \( S \) and \( T \) if and only if the subgraph, \( G' \), induced by \( V_{1,m} \), has a paired \( k_1 \)-DPC joining \( S_1 \) and \( T_1 \) and moreover, the subgraph, \( G_2 \), induced by \( V_{m+1,n} \), has a paired \( k_2 \)-DPC joining \( S_2 \) and \( T_2 \), where \( S_i = S \cap V(G_i) \), \( T_i = T \cap V(G_i) \), and \( k_i = |S_i| \) for \( i \in \{1, 2\} \).
Fig. 10: (a) There exists a unique paired 2-DPC joining \{s_1, s_2\} and \{t_1, t_2\}; (b) There is a unique paired (or unpaired) 1-DPC, which is a hamiltonian \(s_1-t_1\) path.

**Proof.** The sufficiency proof is obvious. To show necessity, we assume (by the counterpart of Lemma 10) that for some \(m\), where \(q' \leq m < p'\), the subgraph, \(G_1\), induced by \(V_{l,m}\), has a paired \(k_1\)-DPC joining \(S_1\) and \(T_1\) and moreover, the subgraph, \(G_2\), induced by \(V_{m+1,n}\), has a paired \(k_2\)-DPC joining \(S_2\) and \(T_2\). By the definition of \(m^*\) and Lemma 4(b), we conclude that subgraph \(G_1\) has a paired \(k_1\)-DPC joining \(S_1\) and \(T_1\), and furthermore \(m \leq m^*\). Also, \(G_2\) has a paired \(k_2\)-DPC joining \(S_2\) and \(T_2\) by Lemma 5(a). Therefore, the proof is complete.

**Remark 8.** To determine \(m^*\) of Eq. 2, we need to check whether or not there exist paired \(k_1\)-DPCs in at most two subgraphs, \(G[V_{q',q'+1}]\) and \(G[V_{q',q'+2}]\). The maximum \(j\) such that \(G[V_{q'+1,j}]\) has a hamiltonian \(v_{q'+1} \rightarrow v_{q'+2}\) path is computed easily as done in lines 25 through 27 of procedure **Find-Unpaired-DPC**.

There remains a task to design an algorithm, which is a counterpart of procedure **Find-UDPC-A**, for determining the existence of a paired DPC in the induced subgraph \(G[V_{l,r}]\) as well as building a paired DPC, if any, where \(v_p\) and \(v_q\), respectively, are the leftmost source and the rightmost sink, and moreover, \(h_j \geq 1\) for all \(p \leq j < q\). Let \(H\) denote the induced subgraph \(G[V_{l,r}]\), and let \(S_H = S \cap V(H)\), \(T_H = T \cap V(H)\), and \(k_H = |S_H| = |T_H|\). If it is clear from the context that we are dealing with the subgraph \(H\), we use \(S, T\) and \(k\), respectively, in order to denote \(S_H, T_H\) and \(k_H\) for notational simplicity.

If there are two valleys, one left \(v_a \rightarrow v_b\) valley and one right \(v_c \rightarrow v_d\) valley, such that the mate of an end-vertex, say \(v_b\), of the left valley is an end-vertex, say \(v_c\), of the right valley, then the two valleys may be combined into a single one, named a **wide** \(v_a \rightarrow v_d\) valley, via an edge \((v_b, v_c)\). Here, \(v_a\) and \(v_d\) are the end-vertices of the wide valley. As illustrated in Fig. 10, a wide valley is not always avoidable when we construct a paired DPC (and also not always avoidable to construct an unpaired 1-DPC, although we think little of the existence of a wide valley when we design an unpaired DPC algorithm in Section 5).

First of all, we study the problem of determining and finding, if any, a paired \(k\)-DPC having a wide valley in \(H\). Note that a paired \(k\)-DPC with a wide valley exists only if there exists an edge between \(V_{l,p-1}\) and \(V_{q+1,r}\). Thus, we assume \(V_{l,p-1}, V_{q+1,r} \neq \emptyset\) and \((v_{p-1}, v_{q+1}) \in E(H)\). It follows that the induced subgraph \(G[V_{p,q}]\) forms a clique by Lemma 1. Let the unique left and right valley, respectively, be \(v_{a} \rightarrow v_{b}\) valley and \(v_{c} \rightarrow v_{d}\) valley in the shape of Lemma 3 where \(a \leq b\) and \(c \leq d\). Then, there are at most four candidates for wide valleys: \(v_{a} \rightarrow v_{c}, v_{b} \rightarrow v_{d}, v_{b} \rightarrow v_{c}\), and \(v_{a} \rightarrow v_{d}\) valleys. It can be determined easily whether or not there exists a paired DPC that contains a wide valley as follows:
Lemma 13. The subgraph $H$ has a paired $k$-DPC with a wide valley if and only if there exists a wide $v_a-v_d$ valley in $H$ and there exist distinct vertices $x \in N_H(v_a) \cap V_{p,q}$ and $y \in N_H(v_d) \cap V_{p,q}$ such that

1. at least one of $x$ and $y$ is a non-terminal, or
2. $\{x, y\} = \{s_i, t_i\}$ for some $i$ and moreover either $k \geq 2$ or $k = 1$ and $V_{p,q} = \{s_i, t_i\}$.

Proof. Necessity. Suppose that $H$ has a paired $k$-DPC with a wide $\alpha-\beta$ valley, where $\alpha \in \{v_a, v_b\}$ and $\beta \in \{v_c, v_d\}$. If $\alpha \neq v_a$, then for the mates $v_a'$ and $v_b'$ of $v_a$ and $v_b$, respectively, we have $(v_a, v_b), (v_b, v_a') \in E(H)$ by the existence of a wide valley. Thus, replacing $(v_a, v_a')$ and $(v_b, v_b')$ with $(v_a', v_b)$, $(v_b', v_a)$ from the DPC results in a paired $k$-DPC with a wide $v_a-\beta$ valley. Similarly, we can obtain a paired $k$-DPC with a wide $v_a-v_d$ valley if $\beta \neq v_d$, proving the existence of a wide $v_a-v_d$ valley. Let $x$ and $y$ respectively be the mates of $v_a$ and $v_d$. Obviously, at least one of $x$ and $y$ is a non-terminal, or $\{x, y\} = \{s_i, t_i\}$ for some $i$. For the latter case, the $s_i-t_i$ path must be a combination of three paths: one-vertex path $(s_i)$, the wide $v_a-v_d$ valley, and $(t_i)$. Thus, if $k = 1$, there is no vertex in $V_{p,q} \setminus \{s_i, t_i\}$, or equivalently, $V_{p,q} = \{s_i, t_i\}$, proving the necessity.

Sufficiency. We will construct a paired $k$-DPC with a wide $v_a-v_d$ valley. Recall that the induced subgraph $G[V_{p,q}]$ is a clique. There are three cases. First, suppose that $x$ and $y$ are both non-terminals. Let $s_j-t_j$ path for $j \geq 2$ be a two-vertex path $(s_j, t_j)$. There exists a paired 2-DPC composed of $s_1-x$ path and $y-t_1$ path in the subgraph induced by $V_{p,q} \setminus \{s_j, t_j : j \geq 2\}$. To obtain a paired $k$-DPC of $H$, it suffices to combine the three paths, $s_1-x$ path, the wide $v_a-v_d$ valley, and $y-t_1$ path, via edges $(x, v_a)$ and $(v_d, y)$ into an $s_1-t_1$ path. Second, suppose that one of $x$ and $y$, say $y$, is a non-terminal. Assume w.l.o.g. $x = s_1$. If we regard $y$ as a virtual source, there exists a paired $k$-DPC composed $y-t_1$ path and $s_j-t_j$ path for $j \geq 2$ in the subgraph induced by $V_{p,q} \setminus s_1$. It suffices to combine the three paths, one-vertex path $(s_1)$, the wide $v_a-v_d$ valley, and the $y-t_1$ path, into an $s_1-t_1$ path. Third, suppose that $\{x, y\} = \{s_i, t_i\}$ for some $i$ and moreover, either $k \geq 2$ or $k = 1$ and $V_{p,q} = \{s_i, t_i\}$. The three paths, one-vertex path $(s_i)$, the wide $v_a-v_d$ valley, and $(t_i)$, are combined into an $s_i-t_i$ path. If $k = 1$ and $V_{p,q} = \{s_1, t_1\}$, we are done; Otherwise, it suffices to find a paired $(k-1)$-DPC in the subgraph induced by $V_{p,q} \setminus \{s_i, t_i\}$, completing the proof.

Lemma 14. The problem of finding a paired $k$-DPC having a wide valley on the subgraph $H$ can be solved in time polynomial to the number of vertices in $H$.

Proof. We can determine the existence of a wide $v_a-v_d$ valley and of a pair of distinct vertices $x$ and $y$ of Lemma 13 in time polynomial to $|V(H)|$. Moreover, the paired $k$-DPC can be constructed, if both exist, in linear-time to $|V(H)|$ according to the sufficiency proof of Lemma 13 and so the lemma follows.

Hereafter, we then study the problem of determining and finding, if any, a paired $k$-DPC without a wide valley. We assume w.l.o.g. that there exists no edge between $V_{l,p-1}$ and $V_{q+1,r}$. (For our purpose, every such edge of $H$ may be removed.) To overcome the trouble due to the absence of counterparts of Lemmas 6 and 8, we will reduce our problem to the PAIRED $k$-DPC problem on an acyclic digraph. A paired $k$-DPC in a digraph naturally refers to a disjoint path cover composed of $k$ directed paths, each starting at a source and ending at its designated sink.

Let us consider the case when $p = l$ and $q = r$, first, where $v_l$ is a source and $v_r$ is a sink.

Lemma 15. Suppose that $p = l$, $q = r$, and $h_j \geq 1$ for all $p \leq j < q$. If the graph $H$ has a paired $k$-DPC joining $S$ and $T$, then $H$ has a paired $k$-DPC joining $S$ and $T$ whose paths are all monotone.
Theorem 4. Suppose there exists a paired $k$-DPC in which not every path is monotone. We first claim that for each $s_i$-$t_i$ path, $P_i$, in the DPC, there exists a monotone $s_i$-$t_i$ path $P'_i$ such that $V(P'_i) \subseteq V(P_i)$. The path $P'_i$ can be built in a greedy manner, as follows: Let $P_i = (w_1, \ldots, w_m)$, where $w_1 = s_i$ and $w_m = t_i$, and let $P'_i$ be empty initially; For each vertex $w_j$ encountered when we traverse $P_i$ from $w_1$ to $w_m$, we pick up $w_j$ and append it to $P'_i$ if and only if (i) $w_j$ is a terminal or (ii) $w_j$ is to the left of $t_i$ and to the right of the latest vertex, $w_f$, appended to $P'_i$. It remains to show that for each $j \geq 2$, $(w_f, w_j) \in E(H)$ if $w_j$ is picked up and appended to $P'_i$ by the greedy algorithm. Suppose $j' < j-1$ as the case of $j' = j-1$ is obvious. Note that $X := \{w_{j+1}, \ldots, w_{j-1}\}$ is nonempty and no vertex of $X$ is chosen by the algorithm. It follows that every vertex $x \in X$ is a non-terminal and moreover, it is either to the right of $t_i$ or to the left of $w_f$. If $w_{j-1}$ is to the left of the $w_f$, we can conclude $(w_f, w_j) \in E(H)$ by Lemma 1. Otherwise, there exists a vertex $w_{j'} \in X$ to the right of $t_i$ such that $w_{j'-1} = w_{j'}$ or $w_{j'-1}$ is to the left of $w_f$, which implies $(w_f, w_j) \in E(H)$ by Lemma 1 again. Thus, the claim is proved.

Now, we have a set of $k$ pairwise disjoint paths $\{P'_1, \ldots, P'_k\}$, in which each path $P'_i$ joining $s_i$ and $t_i$ is monotone. If there exists a (non-terminal) vertex $w$ not covered by any of the $k$ monotone paths, we pick up a path $P'_i$ such that $s_i$ is to the left of $w$ and $t_i$ is to the right of $w$. This is possible because $h_1 \geq 1$ for every $p \leq j < q$. Then, there is an edge, $(x, y)$, of $P'_i$ such that $x$ is to the left of $w$ and $y$ is to the right of $w$. If we replace the subpath $(x, y)$ of $P'_i$ with a new subpath $(x, w, y)$, we have a new monotone path $P''_i$ that joins $s_i$ and $t_i$. Note that $(x, w), (w, y) \in E(H)$ by Lemma 1. Repeating this process eventually produces a paired $k$-DPC whose paths are all monotone. This completes the proof.

The above lemma suggests that the existence of a paired DPC, whose paths are all monotone, is linked directly to the paired DPC problem on an acyclic digraph. That is, under the condition that $p = l$, $q = r$, and $h_1 \geq 1$ for all $p \leq j < q$, there exists a paired $k$-DPC in $H$ if and only if there exists a paired $k$-DPC in an acyclic digraph $D$.

\begin{itemize}
  \item whose vertex set $V(D) = V(H)$ and
  \item arc set $E(D) = \{(v_i, v_j) : (v_i, v_j) \in E(H) \text{ and } v_j \text{ is to the left of } v_i\}$.
\end{itemize}

On the other hand, there exists no paired DPC whose paths are all monotone in the remaining case where $p \neq l$ or $q \neq r$, due to the existence of a valley, left or right. However, there exists a paired DPC, if any, whose paths are good from a monotonicity point of view, as discussed in Lemma 16 below.

Lemma 16. Suppose that $l < p$ or $q < r$, and $h_1 \geq 1$ for all $p \leq j < q$. Let there exist no edge between $V_{l,p-1}$ and $V_{q+1,r}$. If the graph $H$ has a paired $k$-DPC whose valley(s) are in the shape of Lemma 3, then $H$ has a paired $k$-DPC $\mathcal{P}$ such that the valley(s) of $\mathcal{P}$ retain the shape of Lemma 3 and moreover, the set of $k$ paths obtained from $\mathcal{P}$ by the valley removal process below becomes a paired $k$-DPC of the induced subgraph $G[V_{p,q}]$, whose paths are all monotone:

\begin{itemize}
  \item If a path $P_j$ in the DPC contains a left valley, then we remove the left valley from $P_j$ and add an edge to $P_j$ that joins the two mates of the end-vertices of the left valley;
  \item If a path $P_j$ in the DPC, possibly $j = i$, contains a right valley, then we remove the right valley from $P_j$ and add an edge to $P_j$ that joins the two mates of the end-vertices of the right valley.
\end{itemize}

Proof. Suppose there exists a paired $k$-DPC $\mathcal{P}'$ in $H$ whose valley(s) are in the shape of Lemma 3. If $\mathcal{P}'$ has a left valley, let the valley be a $v_{a'}-v_b$ valley with $a \leq b$ and let $v_{a'}$ and $v_{b'}$, respectively,
be the mates of \( v_a \) and \( v_b \) such that \( a' < b' \). Also, if \( P' \) has a right valley, let the valley be a \( v_c-v_d \) valley with \( c \leq d \) and let \( v_{c'} \) and \( v_{d'} \), respectively, be the mates of \( v_c \) and \( v_d \) such that \( c' < d' \). If \( P' \) has both valleys, we assume w.l.o.g. that \( a' < c' \) or \( b' < d' \), so that \( a' < d' \); Suppose otherwise (i.e., \( c' \leq a' < d' < b' \)), it suffices to switch the two valleys, i.e., to set \( v_{c'} \) and \( v_{d'} \), respectively, be the mates of \( v_c \) and \( v_d \), and set \( v_{d'} \) and \( v_{b'} \), respectively, be the mates of \( v_d \) and \( v_b \). Note that the case \( \{a',b'\} = \{c',d'\} \) never occurs; Suppose otherwise, the two valleys and their common mates form a cycle, which contradicts the fact that \( P' \) is a paired DPC. Because the two mates of the end-vertices of a valley, left or right, are adjacent in \( H \), as claimed in the proof of Lemma 15, we can restructure \( P' \) after the valley removal process from \( P' \) becomes a paired \( k \)-DPC of the subgraph \( G[V_{P'_{\alpha}}] \). Suppose not every path in \( Q' \) is monotone. (Suppose otherwise, we are done.) By Lemma 15 we can restructure \( Q' \) into a new paired \( k \)-DPC, \( Q, \) of the subgraph so that all paths in \( Q \) are monotone. Hereafter, we will build a paired \( k \)-DPC, \( P' \), of \( H \) from \( Q \) and the valley(s) of the \( k \)-DPC \( P' \) so that \( P \) after the valley removal process becomes \( Q \).

Suppose \( P' \) has a left \( v_a-v_b \) valley. Let \( v_{c'} \in V(P_i) \) for some \( s_j-t_i \) path, \( P_i, \) in \( Q \). We claim that \( P, \) and the left valley can be combined into a new \( s_j-t_i \) path, \( P_j, \) such that \( P_i \) after the valley removal process becomes \( P_j. \) If \( v_{c'} \neq s_i, \) there exists a predecessor, \( \alpha, \) of \( v_{c'} \) in \( P_i, \) which is the vertex encountered just before \( v_{c'} \) when we traverse \( P_i, \) from \( s_i \) to \( t_i. \) If we combine the two subpaths, the \( s_i-\alpha \) subpath and the \( v_{c'}-t_i \) subpath obtained by removing the edge \((\alpha, v_{c'}) \) from \( P_i, \) with the left valley via edges \((v_{c'}, \alpha) \) and \((v_b, v_{d'}), \) we have a new \( s_j-t_i \) path that passes through the left valley. If \( v_{c'} = s_i, \) there exists a successor, \( \beta, \) of \( v_{c'} \) in \( P_i, \) which is the vertex encountered just after \( v_{c'} \) when we traverse \( P_i, \) from \( s_i \) to \( t_i. \) Then, \( \beta = v_{d'} \) or \( \beta \) is to the left of \( v_{d'} \). (Suppose to the contrary that \( v_{d'} \) is to the left of \( \beta. \) Then, \( v_{d'} \) would be a non-terminal that is to the right of \( v_{c'} \) and to the left of \( t_i. \) Since \((v_{d'}, v_{d'}) \) is an edge of the \( s_j-t_i \) path of \( Q', \) the greedy algorithm for restructuring \( Q' \) into \( Q \) would pick up \( v_{d'} \) and append it to the incomplete \( s_j \)-path as a successor of \( s_i, \) which contradicts the fact that \( \beta \) is the successor of \( s_i \) in \( P_i.\)) If we combine the two subpaths, obtained by removing the edge \((\beta, v_{d'}) \) from \( P_i, \) and the left valley via edges \((v_a, v_{d'}) \) and \((v_b, \beta), \) we have a new \( s_j-t_i \) path that passes through the left valley.

Now, suppose \( P' \) has a right \( v_c-v_d \) valley. Let \( v_{c'} \in V(P_j) \) for some \( s_j-t_i \) path, \( P_j, \) in \( Q, \) and let the \( s_j-t_i \) path \( P_j, \) constructed above, pass through the left valley if \( P' \) has a left valley. If \( P' \) has no left valley or \( j \neq i, \) then we can construct a new \( s_j-t_i \) path, \( P_i, \) that passes through the right valley in a symmetric way to the construction of \( P_j. \) (That is, if \( v_{c'} \neq t_j, \) then the \( s_j-v_{c'} \) subpath, the \( \beta'-t_i \) subpath and the right valley are combined into a new \( s_j-t_i \) path through edges \((v_{c'}, v_{c'}) \) and \((\beta', v_{d'}), \) where \( \beta' \) is the successor of \( v_{d'} \) in \( P_j \); if \( v_{c'} = t_j, \) then the \( s_j-\alpha' \) subpath, one-vertex subpath \((t_j) \) and the right valley are combined into a new \( s_j-t_i \) path through edges \((\alpha', v_c) \) and \((v_{d'}, v_{d'}), \) where \( \alpha' \) is the predecessor of \( v_{d'} \) in \( P_j).\) Let \( j = i \) now. We can see that the aforementioned procedure for processing the right valley successfully combines \( P_i \) and the right valley into a final \( s_j-t_i \) path that passes through both valleys, if the edge \((v_{d'}, \beta') \) of \( P_j \) (resp. the edge \((\alpha', v_{d'}) \) of \( P_i \)) is also an edge of the monotone \( s_j-t_i \) path of \( P_i \) for the case of \( v_{d'} \neq t_i \) (resp. for the case of \( v_{d'} = t_i \) ). We will show that the precondition is satisfied so as to conclude that \( P_j \) and the right valley are successfully combined. From our assumption of \( a' < b', \) the edge \((v_{d'}, \beta') \) of \( P_j \) (for the case of \( v_{d'} \neq t_i \)) is always an edge of the \( s_j-t_i \) path \( P_i \in Q. \) Also, the edge \((\alpha', v_{d'}) \) of \( P_i \) (for the case \( v_{c'} = t_i \)) is an edge of \( P_i \in Q \) if \( v_{d'} \neq s_i, \) or if \( v_{d'} = s_i \) and \( \beta \neq v_{d'}. \) There remains only one possibility of \( v_{c'} = s_i \) and \( \beta = v_{d'} \) \((v_{d'} = t_i), \) It follows that \( P_i \in Q \) is a two-vertex path \((s_i, t_i), \) and that the original \( s_j-t_i \) path of \( P' \) contains both valleys and passes through the other two mates, \( v_a \) and \( v_b, \) of the left and right valleys as intermediate vertices. In the process of restructuring \( Q' \) into \( Q, \) every non-terminal of the \( s_j-t_i \) path of \( Q', \) including \( v_{c'}, \) are not picked up, which implies that every non-terminal of the path is to the left of \( s_i \) or
and whose arc set $P$ to the right of $P$.

We have

\[ \square \]

and $16$.

The following four (see Fig. 11(b)):

\[ \square \]

and $16$.

The following four (see Fig. 11(b)):

\[ \square \]

Based on the above lemma, we can reduce our \textit{Paired $k$-DPC} problem on the graph $H$ into a \textit{Paired $k$-DPC} problem on a (not necessarily acyclic) digraph $D$, whose vertex set $V(D) = V(H)$ and whose arc set $E(D) = C \cup L \cup R$, where $C$, $L$, and $R$ are defined as follows (see Fig. 11(a)):

- $C := \{(v_i, v_j) : (v_i, v_j) \in E(H) \text{ and } v_i \text{ is to the left of } v_j\}$
- $L := \emptyset$ if $l = p$ or there is no hamiltonian $v_{a}-v_b$ path in $G[V_{i,p-1}]$; Otherwise, $L := L_1 \cup L_2 \cup L_3$, where $L_1 := \{(v_i, v_a) : v_i \in N_H(v_a) \cap V_{p,q}\}$, $L_2 := \{(v_b, v_j) : v_j \in N_H(v_b) \cap V_{p,q}\}$, and $L_3 := \{(x_1, x_2), \ldots, (x_{j-1}, x_j)\}$ for a hamiltonian $v_a-v_b$ path, $(x_1, \ldots, x_j)$, of $G[V_{i,p-1}]$.
- $R := \emptyset$ if $r = q$ or there is no hamiltonian $v_{c}-v_d$ path in $G[V_{p+1,r}]$; Otherwise, $R := R_1 \cup R_2 \cup R_3$, where $R_1 := \{(v_j, v_c) : v_j \in N_H(v_c) \cap V_{p,q}\}$, $R_2 := \{(v_d, v_f) : v_f \in N_H(v_d) \cap V_{p,q}\}$, and $R_3 := \{(y_1, y_2), \ldots, (y_{q-1}, y_q)\}$ for a hamiltonian $v_c-v_d$ path, $(y_1, \ldots, y_q)$, of $G[V_{p+1,r}]$.

\textbf{Lemma 17.} Graph $H$ has a paired $k$-DPC without a wide valley if and only if the digraph $D$ has a paired $k$-DPC.

\textbf{Proof.} The sufficiency part is obvious because $D$ is an orientation of a spanning subgraph of $H$, i.e., $(v_i, v_j)$ is an arc of $D$ only if $(v_i, v_j)$ is an edge of $H$. The necessity part is due to Lemmas 15 and 16.

The reduced problem of finding a paired $k$-DPC in the digraph $D$, however, is not an easy task because $D$ may have a cycle. To avoid difficulty, we employ a number of acyclic digraphs $D_{i,j}$ for $i, j \in \{p, \ldots, q-1\}$ (instead of $D$), which is defined in the same way as digraph $D$, except the following four (see Fig. 11(b)):

- $L_1 := \{(v_i, v_a) \text{ if } (v_i, v_a) \in E(H); L_1 := \emptyset \text{ otherwise};$
- $L_2 := \{(v_b, v_f) : v_f \in N_H(v_b) \cap V_{p+1,q}\}$.

Fig. 11: The digraph $D$ and $D_{i,j}$ built from the graph shown in Fig. 11. The dotted lines indicate the absence of arcs.

to the right of $t_i$, particularly, $v_{b'}$ is to the right of $t_i (= v_{a'})$, and $v_{c'}$ is to the left of $s_i (= v_{t'})$. Thus, we have $c' < d' < d' < b'$, which contradicts our assumption of $a' < c'$ or $b' < d'$, completing the entire proof.

\[ \square \]
• $R_1 := \{(v_j, v_c) \mid (v_j, v_c) \in E(H); R_1 := \emptyset$ otherwise;
• $R_2 := \{(v_d, v_f) \mid v_f \in N_H(v_d) \cap V_{j+1,q}\}$.

**Lemma 18.** Graph $H$ has a paired $k$-DPC without a wide valley if and only if there exists a digraph $D_{i,j}$ that has a paired $k$-DPC for $i, j \in \{p, \ldots, q - 1\}$.

**Proof.** The sufficiency part is straightforward by Lemma 17 because every $D_{i,j}$ is a spanning subgraph of $D$. To prove necessity, suppose $H$ has a paired $k$-DPC, $P$, without a wide valley. We assume that the valley(s) of $P$, if any, are in the shape of Lemma 5. We further assume that every path of $P$ is monotone if there is no valley (Lemma 15), or every path obtained from $P$ by the valley removal process is monotone (Lemma 16). If $P$ has a left $v_a \rightarrow v_b$ valley, let $i' = a'$, where $v_a$ is the mate of $v_r$; otherwise, let $i'$ be an arbitrary integer in $\{p, \ldots, q - 1\}$. Similarly, if $P$ has a right $v_c \rightarrow v_d$ valley, let $j' = c'$, where $v_c$ is the mate of $v_r$; otherwise, let $j'$ be an arbitrary integer in $\{p, \ldots, q - 1\}$. Then, the digraph $D_{i',j'}$ has a paired $k$-DPC that corresponds to $P$. □

An algorithm for the Paired $k$-DPC problem on an acyclic digraph will be designed by slightly modifying an algorithm for the $k$-Disjoint Paths or $k$-Linking problem on an acyclic digraph, where $k$-disjoint paths from $S = \{s_1, \ldots, s_k\}$ to $T = \{t_1, \ldots, t_k\}$ in a digraph $D$ is a set of vertex-disjoint paths $\{P_1, \ldots, P_k\}$, in which each $P_i$ is a directed $s_i \rightarrow t_i$ path in $D$. Notice that the $k$-disjoint paths from $S$ to $T$, whose paths altogether cover every vertex of the digraph (thus forms a path cover), is the very paired $k$-DPC joining $S$ and $T$. It has been known that the $k$-Disjoint Paths problem on a general digraph is NP-complete [14] (whereas that problem on a general undirected graph is polynomially solvable for fixed $k \geq 2$ [36, 38]). According to Fortune, Hopcroft, and Wyllie [14], the $k$-Disjoint Paths problem on an acyclic digraph is polynomially solvable when $k$ is not a part of the input. In fact, they proved that for any fixed directed (pattern) graph $G'$, there is a polynomial-time algorithm for deciding if an acyclic digraph $G$ contains a subgraph homeomorphic to $G'$. The algorithm works whether or not the node mapping of $G'$ to $G$ is specified. For our purpose, we quote some part from Bang-Jensen and Gutin [4], which describes an algorithm in plain words for the $k$-Disjoint Paths problem on an acyclic digraph and its correctness as follows:

Let $D = (V, A)$ be acyclic and let $x_1, x_2, \ldots, x_k, y_1, y_2, \ldots, y_k$ be distinct vertices of $D$ for which we wish to find a $k$-linking from $(x_1, x_2, \ldots, x_k)$ to $(y_1, y_2, \ldots, y_k)$. We may assume that $d_{P_i}(x_1) = d_{P_i}(y_1) = 0$ for $i = 1, 2, \ldots, k$, since such arcs play no role in the problem and can therefore be deleted. Form a new digraph $D' = (V', A')$ whose vertex set is the set of all $k$-tuples of distinct vertices of $V$. For any such $k$-tuple $(v_1, v_2, \ldots, v_k)$, there is at least one vertex, say $v_i$, which cannot be reached by any of the other $v_j$ by a path in $D$. (Here we used that $D$ is acyclic.) For each out-neighbour $w$ of $v_i$, such that $w \notin \{v_1, v_2, \ldots, v_k\}$, we let $A'$ contain the arc $(v_1, v_2, \ldots, v_{i-1}, v_r, v_{i+1}, \ldots, v_k) \rightarrow (v_1, v_2, \ldots, v_{i-1}, w, v_{i+1}, \ldots, v_k)$. Only arcs as those described above are in $A'$. We claim that $D'$ has a directed path from the vertex $(x_1, x_2, \ldots, x_k)$ to the vertex $(y_1, y_2, \ldots, y_k)$ if and only if $D$ contains disjoint paths $P_1, P_2, \ldots, P_k$ such that $P_i$ is an $(x_i, y_i)$-path for $i = 1, 2, \ldots, k$. Suppose first that $D'$ has a path $P$ from $(x_1, x_2, \ldots, x_k)$ to $(y_1, y_2, \ldots, y_k)$. By definition, every arc

1. Here, $d_{P_i}(x_1)$ and $d_{P_i}(y_1)$, respectively, denote the in-degree of $x_1$ and the out-degree of $y_1$.
2. An $(x_i, y_i)$-path denotes a path from $x_i$ to $y_i$. 

24
of \( P \) corresponds to one arc in \( D \). Hence we get a collection of paths \( P_1, P_2, \ldots, P_k \) such that \( P_i \) is an \((x_i, y_i)\)-path by letting \( P_i \) contain those arcs that correspond to a shift in the \( i \)th vertex of a \( k \)-tuple, \( i = 1, 2, \ldots, k \).

From the above discussion quoted from \([4]\), we can observe with ease that the number of edges of the path \( P \) from \((x_1, \ldots, x_k)\) to \((y_1, \ldots, y_k)\) in \( D' \) is equal to the total number of edges of paths in the corresponding \( k \)-linking from \((x_1, \ldots, x_k)\) to \((y_1, \ldots, y_k)\) in \( D \). This leads to the fact that there exists a paired \( k \)-DPC from \((x_1, \ldots, x_k)\) to \((y_1, \ldots, y_k)\) in \( D \) if and only if there exists a path of length \( |V(D)| - k \) from \((x_1, \ldots, x_k)\) to \((y_1, \ldots, y_k)\) in \( D' \). The condition is equivalent to that the length of a longest path from \((x_1, \ldots, x_k)\) to \((y_1, \ldots, y_k)\) in \( D' \) is \( |V(D)| - k \). The longest path in \( D' \) is computable in time linear to the size of \( D' \) because \( D' \) is also an acyclic digraph. Thus, we have the following:

**Lemma 19.** The Paired \( k \)-DPC problem on an acyclic digraph is polynomially solvable for any fixed \( k \).

Lemmas \([14, 18, 19]\) lead to the following:

**Lemma 20.** The Paired \( k \)-DPC problem on the graph \( H \) is polynomially solvable for any fixed \( k \).

From the discussions so far in this section, especially in Lemmas \([12, 20]\) and Remark \([8]\), we conclude that:

**Theorem 8.** The Paired \( k \)-DPC problem on a unit interval graph is polynomially solvable for any fixed \( k \).

An \( O(n^{k+3}) \)-time and \( O(n^{\max(k,2)}) \)-space implementation of the Paired \( k \)-DPC algorithm discussed in this section (along with some running examples) may be downloaded from [http://tcs.catholic.ac.kr/~jhpark/papers/PDPC_in_UIG.zip](http://tcs.catholic.ac.kr/~jhpark/papers/PDPC_in_UIG.zip). The subproblem of determining and building a paired \( k_i \)-DPC, if any, in the induced subgraph \( H \) of order \( n_i \), where \( k_i = |S \cap V(H_i)| \), can be solved in \( O(n_i) \) time for the DPC with a wide valley and in \( O(n_i^{k_i+3}) \) time for the DPC without a wide valley. The without-a-wide-valley problem is reduced to \( O(n_i^3) \) DPC problems on acyclic digraphs \( D_{k_i,j} \), each of which is reduced again to the longest path problem on some acyclic digraph \( D'_{k_i,j} \), that has an \( O(n_i^{k_i+1}) \)-time solution. If there are \( r, r \geq 2 \), subproblems on \( H_1 \) for \((n_1, k_1)\), \ldots, and on \( H_r \) for \((n_r, k_r)\), where \( n_1 + \cdots + n_r = n \) and \( k_1 + \cdots + k_r = k \), the total running time is \( O(n^{k+3}) \) since \((n_1 + 1)^{k_1+3} + \cdots + (n_r + 1)^{k_r+3} < n^{k+3} \). (Note that \((n_1 + 1)^{k_1+3} + (n_2 + 1)^{k_2+3} < (n_1 + n_2)^{k_1+k_2+2} + (n_1 + n_2)^{k_1+k_2+2} = 2 \cdot (n_1 + n_2)^{k_1+k_2+2} < (n_1 + n_2)^{k_1+k_2+3} \).

The interested readers may refer to the source code for details.

There seems to be much room for improvement in our Paired \( k \)-DPC algorithm. It is open for developing an efficient algorithm for finding a paired \( k \)-DPC in a unit interval graph, whose running time is polynomial in both \( n \) and \( k \). We hope our naive algorithm could initiate future research.

**Acknowledgement**

This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (grant nos. 2012R1A1A2005511 and NRF-2013R1A1A2012890).
References


