Discrete Applied Mathematics 205 (2016) 132–149

Contents lists available at ScienceDirect

Discrete Applied Mathematics

journal homepage: www.elsevier.com/locate/dam

Algorithms for finding disjoint path covers in unit interval graphs

Jung-Heum Park [a], Joonsoo Choi [b], Hyeong-Seok Lim [c,*]

[a] School of Computer Science and Information Engineering, The Catholic University of Korea, Republic of Korea
[b] School of Computer Science, Kookmin University, Republic of Korea
[c] School of Electronics and Computer Engineering, Chonnam National University, Republic of Korea

ARTICLE INFO

Article history:
Received 2 June 2014
Received in revised form 10 September 2015
Accepted 1 December 2015
Available online 19 January 2016

Keywords:
Disjoint path
Path cover
Path partition
Proper interval graph
Graph algorithm

ABSTRACT

A many-to-many k-*disjoint path cover* (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.

1. Introduction

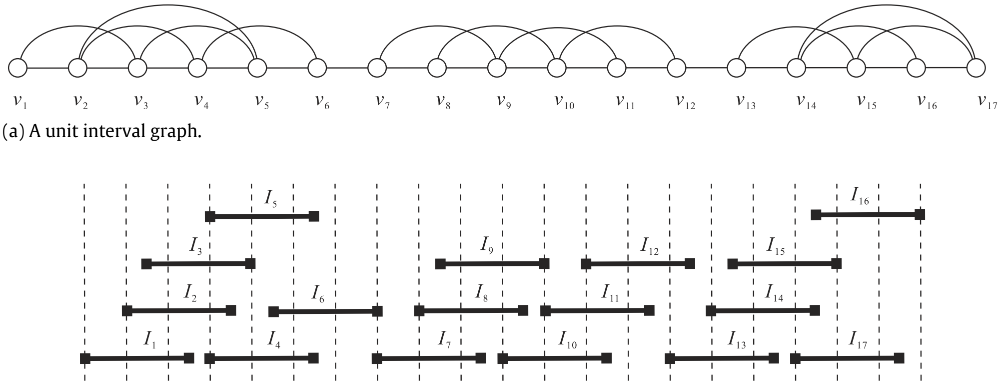
Let G be a simple undirected graph, whose vertex and edge sets are denoted by $V(G)$ and $E(G)$, respectively. A *path cover* of graph G is a set of paths that altogether cover every vertex of G. Of special interest is the *vertex-disjoint path cover*, or simply called *disjoint path cover*, which has 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 [18].

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 $S = \{s_1, \dots, s_k\}$ and a sink set $T = \{t_1, \dots, t_k\}$, 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

* Corresponding author.
E-mail addresses: j.h.park@catholic.ac.kr (J.-H. Park), jschoi@kookmin.ac.kr (J. Choi), hslim@chonnam.ac.kr (H.-S. Lim).

http://dx.doi.org/10.1016/j.dam.2015.12.002
0166-218X/



(b) An interval representation of the graph of (a).

Fig. 1. A unit interval graph and its interval representation.

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 [7,8,13,17,19], recursive circulants [21,22], hypercube-like graphs [20,23,32,33], 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 $\mathcal{T}$ of k vertices of G, a *k-fixed endpoint path cover* of G with respect to $\mathcal{T}$ is a set of vertex-disjoint paths that covers the vertices of G, such that the k vertices of $\mathcal{T}$ are all terminals of the paths in the DPC. For details, refer to [2,3].

An *interval graph* is the intersection graph of family $\mathcal{I}$ of intervals on the real line, where two vertices are connected with an edge if and only if their corresponding intervals intersect. The family $\mathcal{I}$ 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.

An ordering, $(v_1, \ldots, v_n)$, of the vertices of a graph of order n is *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, provided that 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 General-Demand k-DPC [24] problem into the Unpaired k-DPC problem in $O(n + k)$ time, so that the One-to-One DPC and the 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 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., $n = |V(G)|$.

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.*