Theoretical Computer Science ••• (••••) •••–•••

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Solving the maximum internal spanning tree problem on interval graphs in polynomial time

Xingfu Li [a,*], Haodi Feng [b], Haotao Jiang [b], Binhai Zhu [c]

[a] School of Software, Shanxi Agriculture University, Shanxi Provincial, Taigu, Jinzhong, Shanxi, 030801, China
[b] School of Computer Science and Technology, Shandong University, Jinan, Shandong, 250101, China
[c] Department of Computer Science, Montana State University, Bozeman, MT, USA

ARTICLE INFO

Article history:
Received 18 September 2016
Received in revised form 18 July 2017
Accepted 21 September 2017
Available online xxxx

Keywords:
Polynomial
Algorithm
Maximum internal spanning tree
Interval graph

ABSTRACT

This paper studies the Maximum Internal Spanning Tree problem which is to find a spanning tree with the maximum number of internal vertices on a graph. We prove that the problem can be solved in polynomial time on interval graphs. The idea is based on the observation that the number of internal vertices in a maximum internal spanning tree is at most one less than the number of edges in a maximum path cover on any graph. On an interval graph, we present an $O(n^2)$-algorithm to find a spanning tree in which the number of internal vertices is exactly one less than the number of edges in a maximum path cover of the graph, where n is the number of vertices in the interval graph.

1. Introduction

The *Maximum Internal Spanning Tree* problem, MIST briefly, is motivated by the design of cost-efficient communication networks [21]. It asks to find a spanning tree of a graph such that the number of its internal vertices is maximized. Since a Hamilton path (if exists) of a graph is also a spanning tree of that graph with its internal vertices maximized in number, and finding a Hamilton path in a graph is NP-Hard classically [6], MIST is NP-hard trivially. The complement problem of MIST is the so called *Minimum Leaves Spanning Tree* problem, MLST briefly. MLST asks to find a spanning tree of a graph such that the number of its leaves is minimized. One application of MLST appears in water resources engineering [1].

MLST is NP-hard and cannot be approximated to within any constant performance ratio [13], if $P \neq NP$. As algorithmic approaches, Flandrin et al. in [3] and Kyaw in [10] have respectively studied the conditions of whether a graph has a spanning tree with a bounded number of leaves.

Unlike MLST, MIST admits approximation algorithms with a constant performance ratio. Prieto et al. [15] first presented a 2-approximation using local search in 2003. Later, by a slight modification of depth-first search, Salamon et al. [21] improved Prieto's 2-approximation algorithm to running in linear-time. Besides, they proposed a $\frac{3}{2}$-approximation algorithm on claw-free graphs and a $\frac{6}{5}$-approximation algorithm on cubic graphs [21]. Salamon even showed that his 2-approximation algorithm in [21] can achieve a performance ratio $\frac{r+1}{3}$ on r-regular graphs [20]. Furthermore, by local optimization, Salamon [19] devised an $O(n^4)$-time and $\frac{7}{4}$-approximation algorithm on graphs without leaves. Through a different analysis, Knauer et al. [9] showed that Salamon's algorithm in [19] can actually take $O(n^3)$ time to achieve a performance ratio $\frac{5}{3}$ even on general undirected simple graphs. In 2014, Li, Wang and Chen [12] presented a 1.5-approximation algorithm by a local

* Corresponding author.
E-mail addresses: xingfulisdu@qq.com (X. Li), fenghaodi@sdu.edu.cn (H. Feng), htjiang@sdu.edu.cn (H. Jiang), bhz@montana.edu (B. Zhu).

https://doi.org/10.1016/j.tcs.2017.09.017
0304-3975/

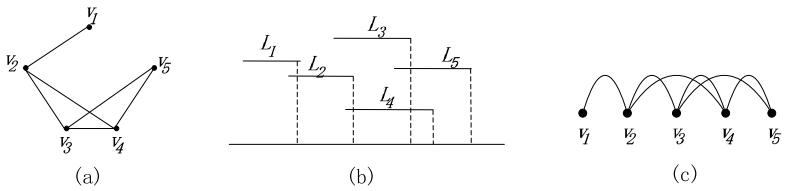

Fig. 1. (a) an interval graph G; (b) an intersection model F of G; (c) the right-end ordering of G.

search method for the maximum internal spanning tree problem on general graphs. Meanwhile, Li and Zhu [11] gave a 1.5-approximation algorithm using a greedy method for the maximum internal spanning tree problem on general graphs.

Salamon et al. [19] also studied the vertex-weighted cases of MIST which asks for a maximum weighted spanning tree of a vertex weighted graph. They gave an $O(n^4)$-time and $(2\Delta-3)$-approximation algorithm for weighted MIST on graphs without leaves, where Δ is the maximum degree of the graph. They also gave an $O(n^4)$-time and 2-approximation algorithm for weighted MIST on claw-free graphs without leaves. Later, Knauer et al. [9] presented a $(3+\epsilon)$-approximation algorithm for weighted MIST on undirected simple graphs.

Fixed parameter algorithms of MIST have also been extensively studied in the recent years. Prieto and Sloper [15] designed the first FPT-algorithm with running time $O^*(2^{4klogk})$ in 2003. Coben et al. [2] improved this algorithm to achieve a time complexity $O^*(49.4^k)$. Then an FPT-algorithm for MIST with time complexity $O^*(8^k)$ was proposed by Fomin et al. [5], who also gave an FPT-algorithm for its directed version with time complexity $O^*(16^{k+o(k)})$ [4]. For directed graphs, a randomized FPT algorithm proposed by M. Zehavi is by now the fastest one which runs in $O^*(2^{(2-\frac{\Delta+1}{\Delta(\Delta-1)})k})$ time [22], where Δ is the vertex degree bound of a graph. On cubic graphs in which each vertex has degree three, Binkele-Raible et al. [1] proposed an $O^*(2.1364^k)$ time algorithm.

For the kernalization of MIST, Prieto and Sloper first presented an $O(k^3)$-vertex kernel [15,14]. Later, they improved it to $O(k^2)$ [16]. Recently, Fomin et al. [5] gave a $3k$-vertex kernel for this problem, which is the best by now.

As for the exact exponential algorithms to solve MIST, Binkele-Raible et al. [1] proposed a dynamic programming algorithm with time complexity $O^*(2^n)$. On graphs with maximum vertex degree bounded by 3 especially, they devised a branching algorithm of $O(1.8612^n)$ time and polynomial space.

In this paper, we pay attention to the maximum internal spanning tree problem on interval graphs. Interval graphs have received a lot of attention due to their applicability to DNA physical mapping problems [7], and find many applications in several fields and disciplines such as genetics, molecular biology, scheduling, VLSI circuit design, archaeology and psychology [8]. We prove that there is a polynomial algorithm to find a maximum internal spanning tree on interval graphs. We present an algorithm with time complexity $O(n^2)$, where n is the number of vertices in an interval graph.

This paper is organized as follows. Section 2 presents basic notations and properties of interval graphs. In Section 3, we present our polynomial algorithm and its analysis. Section 4 concludes this paper by looking forward to some future work on the maximum internal spanning tree problem.

2. Preliminaries

In this paper, all graphs in which we are going to find spanning trees are undirected, simple and connected. Each path or cycle in a graph is always simple. A path (cycle) is simple if it has no repeating vertices. The first and the last vertices of a path are the *endpoints* of that path, while the others except the endpoints of a path are the *inner* vertices of that path. The *length* of a path or cycle is the number of edges in it. A *connected component* of a graph is a subgraph in which any two vertices are connected to each other by paths, and which is connected to no additional vertices in the supergraph. A connected component of a graph is referred to as a *path* (*cycle*) *component* if it is a path (cycle) of that graph. A vertex in a graph is a *leaf* if its degree is 1, and *internal* otherwise.

A spanning subgraph of G is a *path cover* if every connected component of it is a path. A path cover of G is *maximum* if its edges are maximized in number over all path covers of G.

A *maximum internal spanning tree* of G is a spanning tree of G whose internal vertices are maximized in number over all spanning trees of G. The *Maximum Internal Spanning Tree* problem, MIST namely, is given by an undirected simple graph, and asks to find a maximum internal spanning tree for that graph.

A graph G is called an *interval graph* if its vertices can be put in a one-to-one correspondence with a family F of intervals on the real line such that two vertices are adjacent in G if and only if the corresponding intervals intersect. F is called an *intersection model* for G. A *right-end ordering* of an interval graph G can be obtained by sorting the intervals of the intersection model of G on their right ends in time $O(|V(G)|+|E(G)|)$ [17]. An ordering of the vertices according to this numbering is found to be quite useful in solving some graph-theoretic problems on interval graphs [17]. Throughout this paper, an interval graph is represented by its right-end ordering graph. Fig. 1 shows an interval graph and its corresponding right-end ordering graph.

Lemma 1 *(Ramadingam and Rangan [17]). In a right-end ordering graph of an interval graph, for every three indices i, j, k, if $i<j<k$ and there is an edge between v_i and v_k, then there must be an edge between v_j and v_k.*

Please cite this article in press as: X. Li et al., Solving the maximum internal spanning tree problem on interval graphs in polynomial time, Theoret. Comput. Sci. (2017), https://doi.org/10.1016/j.tcs.2017.09.017