A Linear-time Algorithm to Compute a MAD Tree of an Interval Graph

Elias Dahlhaus
Department of Computer Science
University of Bonn
Bonn, Germany

Peter Dankelmann*
School of Mathematical and Statistical Sciences
University of Natal
Durban, South Africa

R. Ravi†
Graduate School of Industrial Administration
Carnegie Mellon University
Pittsburg (PA), USA

Abstract

The average distance of a connected graph G is the average of the distances between all pairs of vertices of G. We present a linear time algorithm that determines, for a given interval graph G, a spanning tree of G with minimum average distance (MAD tree). Such a tree is sometimes referred to as a minimum routing cost spanning tree.

Keywords: Graph Algorithms, Interval Graphs, Spanning Tree.

1 Introduction

The *average distance* $\mu(G)$ of a finite, connected graph $G = (V, E)$ is the average over all unordered pairs of vertices of the distances,

$$\mu(G) = \binom{|V|}{2}^{-1} \sum_{\{u,v\} \subset V(G)} d_G(u, v),$$

*Financial support by the South African National Research Foundation is gratefully acknowledged.

†This material is based upon work supported by the National Science Foundation under grants CCR 0105548 and ITR 0122581 (ALADDIN Center).

where $d_G(u, v)$ denotes the distance between the vertices u and v. A *minimum average distance spanning tree* of G (*MAD tree* in short) is a spanning tree of G of minimum average distance. MAD trees, also referred to as minimum routing cost spanning trees, are of interest in the design of communication networks [6]. One is interested in designing a tree subnetwork of a given network, such that on average, one can reach every node from every other node as fast as possible. In general, the problem of finding a MAD tree is NP-hard [6]. A polynomial-time approximation scheme is due to [1]. Hence it is natural to ask for which restricted graph classes a MAD tree can be found in polynomial time. In [3], an algorithm is exhibited that computes a MAD tree of a given distance-hereditary graph in linear time. In [5] it is shown that a MAD tree of a given outerplanar graph can be found in polynomial time.

In this paper, we show that for a given interval graph G a MAD tree can be computed in time $O(|E|)$. If the interval representation of G is known and the left and right boundaries of the intervals are ordered, then a MAD tree of G can be found in time $O(|V|)$. In section 2, we present some structural results on MAD trees. In section 3, we sketch the algorithm, and in section 4, we give some concluding remarks.

2 Structure of MAD Trees in Interval Graphs

We always assume that the graph under consideration is connected. The *distance* of a vertex v, $d_G(v)$, is defined as $\sum_{x\in V} d_G(v, x)$. The *total distance* of a graph H denoted $d(H)$ is the sum of all pairwise distances between nodes in H, i.e. $d(H) = \sum_{\{u,v\}\subset V(H)} d_H(u, v)$.

A *median vertex* of G is a vertex c for which $d_G(c)$ is minimum. The *eccentricity* of a vertex v is defined as $ex_G(v) = \max_{w\in V} d_G(v, w)$.

The neighbourhood of a vertex v in G is defined in two ways: the *open* version $N_G(v) = \{u : uv \in E\}$ and the *closed* version $N_G[v] = \{v\} \cup N_G(v)$.

The following lemma applies not only to interval graphs but to all connected graphs. Part (i) improves on a result in [2], which states that, if T is a MAD tree of a given connected graph G, then there exists a vertex c in T such that every path in T starting at c is induced in G. We now prove that c can be chosen to be a median vertex of T.

Lemma 1 *(i) If T is a MAD tree of G and c is a median vertex of T then every T-path starting at c is an induced path in G (i.e. has no diagonals in G).*
(ii) T and c can be chosen such that there is no vertex $c' \neq c$ such that $N_G[c]$ is strictly contained in $N_G[c']$.

Proof: (i) Let $P = v_1, \ldots, v_k$ be a path in T with $c = v_1$. Suppose P is not induced in G. Then there are i and j, such that $i < j - 1$ and $v_iv_j \in E(G)$. Let T_1 and T_2 be the connected components of $T - v_{j-1}v_j$ containing v_{j-1} and v_j, respectively. It is known (see [2]) that $d_T(c) \leq d_T(v_i) < d_T(v_{j-1})$. Since the vertices in T_2 are further away (in T) from v_i than from v_{j-1}, we obtain

$$d_{T_1}(v_i) < d_{T_1}(v_{j-1}).$$

Consider the tree

$$T' = T - v_{j-1}v_j + v_iv_j.$$

Since the distances between any two vertices are the same in T and T', unless one vertex is in T_1 and the other vertex is in T_2, we have

$$d(T') - d(T) = |V(T_2)|(d_{T_1}(v_i) - d_{T_1}(v_{j-1})) < 0,$$

contradicting the minimality of $d(T)$. Hence P is an induced path in G.

(ii) If there is a vertex c' with $N_G[c] \subset N_G[c']$ and $N_G[c] \neq N_G[c']$, then joining all T-neighbours of c to c' instead of c yields a spanning tree T' with $d(T') \leq d(T)$, in which c is an end vertex and c' is a median vertex. QED

From now on we assume that G is an interval graph. Each vertex v corresponds to an interval $I(v) = [l(v), r(v)]$, such that two vertices v and w are adjacent if and only if $I(v) \cap I(w) \neq \emptyset$.

Proposition 1 *Let $v_0, \ldots, v_k$ be an induced path of the interval graph G. Then*

1. *$l(v_1) < l(v_2) < \ldots < l(v_k)$ and $r(v_0) < \ldots < r(v_{k-1})$ or*
2. *$r(v_1) > \ldots > r(v_k)$ and $l(v_0) > \ldots > l(v_{k-1})$.*

In the first case, we call such a path an R-path, in the second case, we call it an L-path.

Consequently, each path of the MAD tree T of G starting at a median vertex c of T is an L-path or an R-path.

For a vertex v of G let $h(v)$ be a neighbour x of v such that $r(x)$ is maximum; If $r(v) = \max_{w \in V} r(w)$, we say $h(v)$ is undefined. Similarly, let $k(v)$ be a neighbour y of v such that $l(y)$ is minimum. Again, if $l(v) = \min_{w \in V} l(w)$, we say that $k(v)$ is undefined. We also define $h^0(v) = k^0(v) = v$ for all v. For $i \geq 2$, $h^i(v)$ is defined as $h(h^{i-1}(v))$. Analogously, we define $k^i(v)$.

A component of a graph is *trivial* if it contains only one vertex, otherwise it is called nontrivial.

For a given vertex v of G and an integer $i \geq 2$ let $V_i^R(v)$ ($V_i^L(v)$) be the set of all vertices at distance i from v, whose intervals lie completely to the right (left) of the interval of v. We also let $V_1^R(v) = V_1^L(v) = N_G(v)$.

Theorem 1 *If G is an interval graph then there is a MAD tree T of G with a median vertex c, such that for each i, $1 \leq i \leq ex_G(c)$,*

$$(*) \begin{cases} \textit{each } v \in V_i^R(c) \textit{ is adjacent in } T \textit{ to } h^{i-1}(c) \\ \textit{each } v \in V_i^L(c) \textit{ is adjacent in } T \textit{ to } k^{i-1}(c). \end{cases}$$

Proof. Let T be a MAD tree of G and let c be a median vertex of T, where c is chosen according to Lemma 1(ii).

By Lemma 1(i), each G-neighbour v of c is also in T adjacent to c, since otherwise the $c - v$ path in T would have a diagonal. Hence $(*)$ holds for $i = 1$.

Suppose that $(*)$ does not hold for some i. Let i be the smallest such number. For the rest of this proof, we omit the reference to the median vertex c in the notation for V^R's.

We first show that only one vertex in V^R_{i-1} has T-neighbours in V^R_i. Suppose that there exist vertices $v_i, v'_i \in V^R_i$, $v_{i-1} \neq v'_{i-1} \in V^R_{i-1}$ such that $v_i v_{i-1}, v'_i v'_{i-1} \in E(T)$. Since, by the minimality of i, v_{i-1} and v'_{i-1} are adjacent in T to $h^{i-2}(c)$, we have $v_i \neq v'_i$ (Else there would be a cycle in T). Without loss of generality, we may assume that $r(v_{i-1}) \geq r(v'_{i-1})$. Let B be the component of $T - v_{i-1}v_i$ not containing c and let B' be the component of $T - h^{i-2}(c)v'_{i-1}$ not containing c. Moreover let

$$child(v'_{i-1}) = \{w \in B' \mid v'_{i-1}w \in E_T\}.$$

Note that $child(v'_{i-1})$ is exactly the set of children of v'_{i-1} when T is rooted at c. Now $child(v'_{i-1}) \subseteq N_G(v_{i-1})$ since every child of v'_{i-1} is on an R-path from c and $r(v_{i-1}) \geq r(v'_{i-1})$. Therefore,

$$T' = T - \{v'_{i-1}w \mid w \in child(v'_{i-1})\} + \{v_{i-1}w \mid w \in child(v'_{i-1})\}$$

is a spanning tree of G. Since the distance between any two vertices are the same in T and T', unless one of the vertices is in B and the other one is in B', the difference between the total distances is

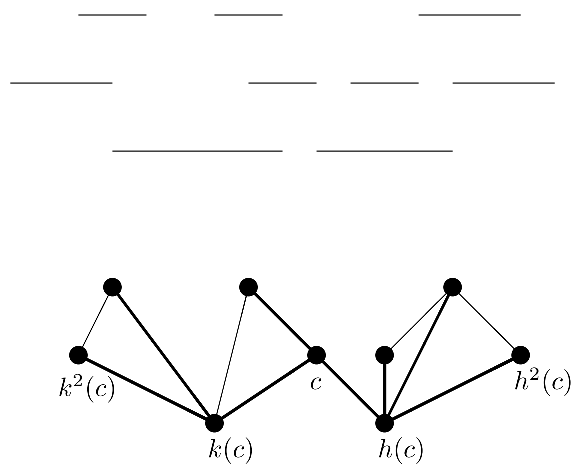
$$d(T') - d(T) = -2|V(B)|(|V(B')| - 1) < 0$$

contradicting the minimality of $d(T)$. Hence at most one vertex v in V^R_{i-1} has T-neighbours in V^R_i. Hence only one vertex in V^R_{i-1} has T-neighbours in V^R_i. Since each vertex in V^R_i that is adjacent in G to a vertex v in V_{i-1} is also adjacent in G to $h^{i-1}(c)$, we can join each vertex in V^R_i not to v, but to $h^{i-1}(c)$ without increasing the total distance of T. Analogously, we can achieve that each vertex in V^L_i is adjacent in T to $k^{i-1}(c)$. Hence T satisfies condition $(*)$. QED

Theorem 1 suggests the following polynomial time algorithm. Fix a vertex c of G, determine $h^i(c)$ and $k^i(c)$ for $i = 1, 2, \ldots$ and construct a spanning tree of G in which each vertex in $V^R_i(c)$ is adjacent to $h^{i-1}(c)$ and each vertex in $V^L_i(c)$ is adjacent to $k^{i-1}(c)$ for $i \geq 1$. Construct such a tree for each $c \in V(G)$. Among those n trees select a tree with minimum total distance. By Theorem 1, this tree is a MAD tree of G.

Theorem 2 *A MAD tree of an interval graph can be determined in polynomial time.*

A set of intervals and the corresponding interval graph G is shown below. The vertices c, $h^i(c)$ and $k^i(c)$, $i = 1, 2$, are labelled. Thick lines indicate the edges of a MAD tree T satisfying the condition $(*)$ of Theorem 1.



3 Linear-time Computation of a MAD Tree

We assume that an interval representation of $G = (V, E)$ is known and that the left borders $l(v)$ and right borders $r(v)$, $v \in V$ are sorted. As in the previous section, we assume that $h(v)$ is a neighbour x of v, such that $r(x)$ is maximum and that $k(v)$ is a neighbour y of v, such that $l(y)$ is minimum. In [7] it is shown that h and k can be determined in time linear in the number of vertices (even in logarithmic time in parallel with a linear workload).

We consider any vertex v and assume that the median vertex c is left (right) of v, i.e., that $r(c) \leq r(v)$ ($l(c) \geq l(v)$).

Let T be a MAD tree according to Theorem 1 rooted at c and let $v \neq c$ be a vertex, $v \in V_i^R(c)$, say. Consider T_v, the subtree of T rooted at v. Then either v is a leaf of T and T_v is trivial, or $v = h^i(c)$ for some i. In that case, T_v contains some G-neighbours of v, in particular $h(v) = h^{i+1}(c)$, and all vertices in $V_{i+2}^R(c) = V_2^R(v)$ (which are in T adjacent to $h^{i+1}(c) = h(v)$), $V_{i+3}^R(c) = V_3^R(v)$ (which are in T adjacent to $h^{i+2}(c) = h^2(v)$), and so on, as long as they are defined. The fact that the main part of T_v, namely $T_{h(v)}$, only depends on whether v is to the left or right of c, but not on the actual choice of c, is the key to our algorithm.

Definition 1 *Let G be a connected interval graph with h and k as defined above, and $v \in V(G)$. If $h(v)$ ($k(v)$) is not defined then let T_v^R (T_v^L) be the empty tree. If $h(v)$ is defined then let T_v^R be the tree with vertex set $\{h(v)\} \cup \bigcup_{j \geq 2} V_j^R(v)$, where a vertex $x \in V_j^R(v)$ is adjacent to $h^{j-1}(v)$. Analogously, if $k(v)$ is defined then let T_v^L be the tree with vertex set $\{k(v)\} \cup \bigcup_{j \geq 2} V_j^L(v)$, where a vertex $x \in V_j^L(v)$ is adjacent to $k^{j-1}(v)$. Note that both T_v^R and T_v^L do* not *contain v!*

Hence the tree $T = T_c$ consists of c as root, the neighbours of c in the original graph G as neighbours in T_c, and T_c^L and T_c^R appended on $k(c)$ and $h(c)$ respectively. We do not determine these trees T_v^R and T_v^L explicitly. Instead, we compute the following quantities.

1. the number of neighbours $num^R(v)$ ($num^L(v)$) of $h(v)$ ($k(v)$) that are not neighbours of v, i.e. the number of children of $h(v)$ ($k(v)$) in T_v^R ($T^L(v)$),
2. $|T_v^R|$ ($|T_v^L|$), the number of vertices of the tree T_v^R (T_v^L),
3. the total distance $t^R(v)$ ($t^L(v)$) of T_v^R (T_v^L), and
4. the distance $d^R(v)$ ($d^L(v)$) of $h(v)$ ($k(v)$) in T_v^R (T_v^L).

The numbers $num^L(v)$ and $num^R(v)$ can be determined overall in $O(n)$ time (see, e.g. [7]). For any particular v, $d^R(v)$ can be determined in $O(1)$ time if $d^R(h(v))$ and $num^R(v)$ are known. Also $t^R(v)$ can be determined in $O(1)$ time if $d^R(v)$, $num^R(v)$ and $t^R(h(v))$ are known. Analogous statements hold for the left counterparts.

In more detail, we proceed as follows.

Determine $num^R(v)$: If $h(v)$ is not defined then $num^R(v) = 0$. Otherwise, $num^R(v)$ is the number of vertices w, such that $r_v < l_w \leq r_{h(v)}$. This can be determined in overall linear time.

Determine $|T_v^R|$: If $h(v)$ is not defined, $|T_v^R| = 0$. Otherwise,

$$|T_v^R| = |T_{h(v)}^R| + num^R(v).$$

Determine $d^R(v)$: If $h(v)$ is not defined then $d^R(v) = 0$. Otherwise

$$d^R(v) = d^R(h(v)) + |T_{h(v)}^R| + num^R(v) - 1.$$

Determine the distance $t^R(v)$: If $h(v)$ is not defined then $t^R(v) = 0$. Otherwise

$$t^R(v) = t^R(h(v)) + d^R(v)$$
$$+(num^R(v) - 1)\Big((num^R(v) - 2) + 2|T^R(h(v))| + d^R(h(v))\Big).$$

Determine the total distance of T_c: First we determine the degree $\delta(c)$ of c. This can be done in overall time $O(n)$, for all c (One has to count the number of right borders between the left border l_c and the right border r_c of c and the number of intervals passing the right border of c).Then we define $n(c)$ to be the total number of neighbours of c excluding itself as well as $h(c)$ and $k(c)$ if they are defined.

The total distance of T_c is

$$(n(c))^2 + t^L(c) + t^R(c) + (1 + 2n(c) + d^L(c))|T^R(c)|$$
$$+(1 + 2n(c) + d^R(c))|T^L(c)| + 2|T_c^L||T_c^R| + (d^L(c) + d^R(c))(1 + n(c)).$$

Thus we compute the numbers $t^R(v)$ $(t^L(v))$, $d^R(v)$ $(d^L(v))$, $|T_v^L|$ $(|T_v^R|)$, and $num^R(v)$ $(num^L(v))$ in overall linear time.

We obtain the total distances and thus the average distances of the trees T_c with assumed median vertices c in overall linear time, and we only have to select one T_c with minimum average distance. We determine this particular T_c explicitly using Theorem 1. Also this can be done in linear time.

Theorem 3 *If an interval representation of an interval graph G with n vertices is given and the left and the right boundaries $l(v)$ and $r(v)$ are sorted then a MAD tree of G can be determined in $O(n)$ time.*

4 Conclusions

It remains an open problem to decide whether there exists a polynomial time algorithm to find a MAD tree of a *vertex weighted* interval graph. If w is a real valued weight function on the vertex set of G, then the average distance of G with respect to w is defined (see [4]) as

$$\binom{w(V)}{2}^{-1} \sum_{\{u,v\} \subset V(G)} w(u)w(v)d_G(x,y),$$

where $w(V)$ is the total weight of the vertices in G.
Theorem 1 does not hold (and hence the algorithm presented above does not work) if there is a weight function on the vertices of G. To see this, consider the path with four vertices $v_1, \ldots, v_4$ of weight $1, 2, 1, 5$ together with a vertex v_5 of weight 0, that is adjacent to $v_1, \ldots, v_4$. An interval representation of this graph is as follows: $v_1 : [0, 1]$, $v_2 : [0, 3]$, $v_3 : [2, 5]$, $v_4 : [4, 5]$, $v_5 : [0, 5]$. This graph has a unique MAD tree, the path v_1, v_2, v_3, v_4, v_5. It is easy to check that T does not contain a vertex satisfying the conclusion of Theorem 1.

New techniques seem necessary to solve the edge-weighted counterpart of the MAD tree problem for Interval graphs. It would also be interesting to know whether our algorithm can be extended to the class of strongly chordal graphs.

References

[1] Bang Ye Wu, G. Lancia, V. Bafna, K.-M. Chao, R. Ravi, Chuang Yi Tang, *A polynomial time approximation scheme for minimum routing cost spanning trees.* SIAM Journal on Computing, Vol. 29, No. 3, pp. 761-778 (1999).

[2] C.A. Barefoot, R.C. Entringer, L.A. Székely, *Extremal Values of Distances in Trees.* Discrete Applied Mathematics 80 (1997), pp. 37-56.

[3] E. Dahlhaus, P. Dankelmann, W. Goddard, H.C. Swart, *MAD trees and distance-hereditary graphs.* Discrete Applied Mathematics 131 (2003), pp. 151-167.

[4] P. Dankelmann, *Computing the average distance of an interval graph.* Inf. Process. Lett. 48 (1993), pp. 311-314.

[5] P. Dankelmann, P. Slater, *Average distance in outerplanar graphs.* Preprint.

[6] D.S. Johnson, J.K. Lenstra and A.H.G. Rinnooy-Kan, *The complexity of the network design problem.* Networks 8(1978), pp. 279–285.

[7] S. Olariu, J. Schwing, J. Zhang, *Optimal Parallel Algorithms for Problems Modeled by a Family of Intervals.* IEEE Transactions on Parallel and Distributed Systems 3 (1992), pp. 364-374.