THE TOTAL INTERVAL NUMBER OF A GRAPH, II: TREES AND COMPLEXITY

Thomas M. Kratzke
Metron, Inc., McLean, VA 22101

Douglas B. West
University of Illinois, Urbana, Illinois 61801

Abstract

A *multiple-interval representation* of a simple graph G assigns each vertex a union of disjoint real intervals, such that vertices are adjacent if and only if their assigned sets intersect. The *total interval number* $I(G)$ is the minimum of the total number of intervals used in any such representation of G. For triangle-free graphs, $I(G) = m + t(G)$, where m is the number of edges in G and $t(G)$ is the minimum number of pairwise edge-disjoint trails such that every edge of G has an endpoint in at least one of the trails. This yields the NP-completeness of testing $I(G) = m + 1$, even for triangle-free 3-regular planar graphs, and an alternative proof that HAMILTONIAN CYCLE is NP-complete for line graphs. It also yields a linear-time algorithm to compute $I(G)$ for trees and a characterization of the trees requiring $m + t$ intervals, for fixed t. Further corollaries include the Aigner/Andreae bound of $I(G) \le \lfloor (5n-3)/4 \rfloor$ for n-vertex trees (achieved by subdividing every edge of a star), a characterization of the extremal trees, and a shorter proof of the extremal bound $\lfloor (5m+2)/4 \rfloor$ for connected graphs.

Keywords: intersection graphs, intervals, trees, extremal problem
AMS Subject Classification: 05C05, 05C35, 05C85
Running head: TOTAL INTERVAL NUMBER, II

[1]Research supported in part by ONR Grant N00014-85K0570.

1. INTRODUCTION

An *intersection representation* of a graph G assigns each vertex v a set $f(v)$ such that u, v are adjacent if and only if $f(u) \cap f(v) \neq \emptyset$. Conversely, the graph is the *intersection graph* of the sets in the representation. The most well-studied class of intersection graphs are the *interval graphs*, which are the intersection graphs obtainable by assigning each vertex a single interval on the real line. More generally, an intersection representation f that assigns each vertex a union of intervals on the real line is a *multiple-interval representation* of G. Let $|f(v)|$ denote the number of (pairwise disjoint) intervals whose union is $f(v)$. If $|f(v)| = k$, then we say that $f(v)$ *consists of* k intervals or that v is *assigned* k intervals.

In two natural ways, multiple-interval representations can measure how far a graph is from being an interval graph. The *interval number* of G is $i(G) = \min_f \max_{v \in V(G)} |f(v)|$, where the minimum is taken over all multiple-interval representations of G. The *total interval number* of G is $I(G) = \min_f \Sigma_{v \in V(G)} |f(v)|$, which can be viewed as minimizing the average number of intervals assigned per vertex instead of the maximum number. Always $I(G) \leq ni(G)$ for n-vertex graphs; the interval graphs without isolated vertices have interval number 1 and total interval number n.

Interval number has been studied for many years, beginning with [10] and [4]. Although introduced in [4], total interval number was not studied until Aigner and Andreae [1] obtained the maximum value of $I(G)$ for several classes of graphs on n vertices, including trees ($\lfloor (5n-3)/4 \rfloor$), 2-connected outerplanar graphs ($\lfloor 3n/2 - 1 \rfloor$), triangle-free planar graphs ($2n-3$), and triangle-free graphs ($\lceil (n^2+1)/4 \rceil$). For the latter three classes, they conjectured that the upper bounds would still hold when the ''2-connected'' or ''triangle-free'' restrictions were removed. In [7], we proved these conjectures for outerplanar and general graphs on n vertices, and we also proved the Aigner-Andreae conjecture that that $\max I(G) = \lfloor (5m+2)/4 \rfloor$ if G is a connected graph with m edges. The proof of their conjecture for planar graphs is quite lengthy and will appear in a later paper in this series. Other papers will study the maximum total interval number for cacti or Husimi trees on n vertices, and for connected graphs with m edges having lower bounds on minimum vertex degree, connectivity, or edge-connectivity. Most of this work appeared in the dissertation of the first author, accepted in 1987 [6].

In this paper, we present a linear-time algorithm to compute the total interval number of a tree (Section 3). This is based on the equality $I(G) = m + t$ for a triangle-free graph with m edges, where t is the minimum number of edge-disjoint trails needed to touch every edge of the graph (Section 2). From this characterization, we also obtain the NP-completeness of testing $I(G) = m + 1$ even for triangle-free 3-regular planar graphs, and an alternative proof of the NP-completeness of HAMILTONIAN CYCLE for line graphs. A closer examination of the algorithm for trees yields a characterization of the trees requiring $m + t$ intervals for fixed t (Section 4). This is turn yields short proofs of the Aigner-Andreae extremal bound for trees and the extremal bound in [7] for connected graphs (Section 5).

2. TRAIL COVERS AND COMPLEXITY

We use n for the number of vertices of a graph G, m for the number of edges, $N(v)$ for the set of neighbors of v, and $x \leftrightarrow y$ for ''x is adjacent to y.''

In studying $I(G)$, we allow $f(v) = \emptyset$, so that isolated vertices contribute nothing to the count of intervals. As in the study of $i(G)$, it is natural to define the *depth* of a representation to be the maximum number of vertices to which a single point is assigned; the *depth-r total interval number* $I_r(G)$ is

the minimum of $\Sigma|f(v)|$ over all representations of G with depth at most r. An interval in $f(v)$ is *displayed* if some portion of it intersects no other interval of f.

A *vertex cover* of G is a set of vertices that contains an endpoint of every edge of G. A collection of pairwise edge-disjoint trails whose vertices together form a vertex cover is a *trail cover*. The *trail cover number* $t(G)$ is the minimum number of trails in a trail cover of G. Traversed from left to right, a depth-2 representation can establish at most one edge for each interval after the first, so $I_2(G) \geq m+1$, and this bound can be achieved only if there are no ''gaps'' in the representation. Lemma 2.1 extends this observation. This lemma appears in [7], but we repeat its proof here because the transformation to trail cover number is essential for computing $I(G)$ for trees. The algorithm of Section 3 constructs a minimum trail cover.

LEMMA 2.1. For a graph G with m edges, $I_2(G) = m + t(G)$, and hence $I(G) = m + t(G)$ if G is triangle-free.

Proof. For triangle-free graphs, $I(G) = I_2(G)$. First we show $I_2(G) \leq m + t(G)$. Let $\{Z_j\}$ be an optimal trail cover. For each trail $Z_j = (v_1, \ldots, v_r)$, choose r intervals in $(j-1, j)$ such that the ith interval intersects only the $i-1$st and $i+1$st intervals (for $2 \leq i \leq r-1$), and add the ith of these intervals to $f(v_i)$. Vertices may appear repeatedly in trails, and all these intervals are displayed. For each edge not in these trails, assign an interval for one endpoint within the displayed portion of its neighbor in $\cup V(Z_j)$. For each trail, the number of intervals used is one more than the number of intervals represented, so we have represented G with $m + t(G)$ intervals.

Conversely, given an optimal depth-2 representation, we obtain a trail cover consisting of $I_2(G) - m$ edge-disjoint trails. Because no more than two intervals intersect at any point, we can eliminate any intersection of intervals by shortening or deleting one interval without affecting any other intersection. Therefore, we may assume that every edge is represented exactly once. Removal of each non-displayed interval from an optimal representation leaves a representation of edge-disjoint trails as described above, having deleted one edge for each interval deleted. Furthermore, the vertices of the resulting trails touch all edges of the original graph. If we now shrink each trail to a single vertex by deleting one interval and edge at a time, we have deleted every edge of G and one interval for each edge. There remain $I_2(G) - m$ intervals, one from each trail in the trail cover. ■

COROLLARY 2.2. The decision problem $I(G) \leq m+1$ is NP-complete, even when restricted to the class of planar, 3-regular, triangle-free graphs.

Proof. The problem is in NP, because it is easy to check whether an assignment of $m+1$ intervals is a representation. For triangle-free graphs, the problem is equivalent to testing whether G has a single covering trail. It is well known that testing for a Hamiltonian path in a 3-regular planar graph is NP-complete [3]. Given an arbitrary 3-regular planar graph G, we replace each vertex by a 7-vertex subgraph as indicated in Figure 1. The resulting graph G' is 3-regular, planar, and triangle-free. It suffices to show that G has a Hamiltonian path if and only if G' has a covering trail. (This transformation was used in [9] to prove the NP-completeness of testing $i(G) \leq 2$.)

Given $v \in V(G)$, let $H(v)$ be the subgraph induced by the seven vertices that replace v in G'. Because $H(v)$ contains edges not incident to vertices of any other $H(w)$, a covering trail must enter every $H(v)$. Since only three edges enter each $H(v)$, a trail can enter and/or exit $H(v)$ only once. Therefore, contracting each $H(v)$ to a single vertex v turns G' into G and a covering trail of G' into a

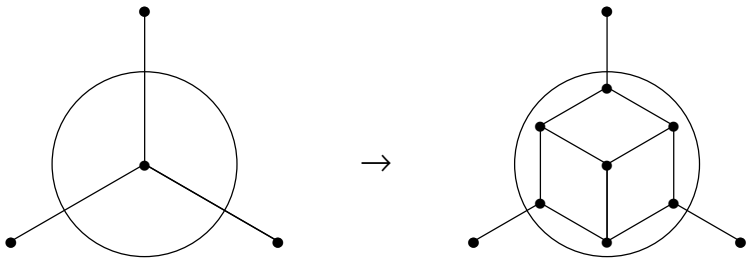

Figure 1. Transformation at each vertex.

Hamiltonian path or cycle in G.

Conversely, if G has a Hamiltonian path P, then we can replace each internal vertex v of the path by a Hamiltonian path in $H(v)$ between the two-valent vertices of $H(v)$ that correspond to the edges incident to v in P. If v is an end of P, we use a Hamiltonian path in $H(v)$ ending at the central vertex of $H(v)$. The result is a Hamiltonian path of G', which is certainly a covering trail. ■

From the Hamiltonian cycle in 3-regular planar graphs, the same transformation proves that it is NP-hard to test whether a graph has a single *closed* covering trail. By combining this with known results about line graphs, we obtain a short alternative proof of the known result that testing for Hamiltonian cycles is NP-hard even when the input is restricted to line graphs. Note that the existence of a Hamiltonian cycle in a line graph $L(H)$ is not equivalent to the existence of an Eulerian circuit in H.

COROLLARY 2.3. (Bertossi [2]) HAMILTONIAN CYCLE is NP-hard on line graphs.

Proof. Given a line graph G with at least four vertices, we can retrieve the unique graph H such that $G = L(H)$ in linear time (Lehot [8]). We also know that G is Hamiltonian if and only if H has a closed covering trail (Harary and Nash-Williams [5]). As observed above, this is NP-hard. ■

3. TRAIL COVERS OF TREES

Our recursive algorithm for computing the trail cover number of a tree computes additional information about the tree. We use (T, x) to denote a tree T with a vertex x distinguished as its *root*. Suppose $v \in V(T)$ and **C** is a trail cover **C** of T. We say that **C** *visits* v if v is a vertex of a trail in **C**, that **C** *ends at* v if v is an endpoint of a trail in **C**, and that **C** *isolates* v if v is a trail of length 0 in **C** (a *degenerate* trail). Isolating v implies ending at v, which in turn implies visiting v. Given a tree T rooted at x, the *code* $c(T, x)$ indicates the most restrictive of these conditions at the root that can be satisfied by a minimum trail cover.

$c(T, x)$	condition
0	no minimum cover visits x
1	some minimum cover visits x but none ends at x
2	some minimum cover ends at x but none isolates x
3	some minimum cover isolates x

We will show that the following recursive algorithm computes the trail cover number and code of a rooted tree.

ALGORITHM 3.1. Input (T, x). Output trail cover number t, code c, and trail cover $\mathbf{C}$ establishing t and c.

If $n(T) = 1$, set $t = 0$, $c = 0$, and $\mathbf{C} = \varnothing$. Otherwise, let $x_1, \ldots, x_k$ be the neighbors of x, designated as roots of the components $T_1, \ldots, T_k$ of $T - x$. Let $t_i, c_i, \mathbf{C}_i$ be the output of the algorithm for T_i rooted at x_i. For $0 \le j \le 3$, let $k_j = |\{i\colon c_i = j\}|$. Note that $\Sigma k_i = k > 0$.

If $k_2 + k_3 = 0$ and $k_1 = k$, then set $\mathbf{C} = \cup\mathbf{C}_i$, $t = \Sigma t_i$, and $c = 0$.

If $k_2 + k_3 = 0$ and $k_1 < k$, then set $\mathbf{C} = (\cup\mathbf{C}_i)\cup\{(x)\}$, $t = 1 + \Sigma t_i$, and $c = 3$.

If $k_2 + k_3 > 0$, then form $\mathbf{C}$ by beginning with $\cup\mathbf{C}_i$ and iteratively joining pairs of trails that end in $\{x_i\colon c_i \ge 2\}$ by edges from those roots to x. Set $t = \Sigma t_i - \lfloor (k_2 + k_3)/2 \rfloor$, and set $c = 1$ if $k_2 + k_3$ is even and $c = 2$ if $k_2 + k_3$ is odd.

Since the computation of c and t uses only $\{c_i\}$ and $\{t_i\}$, the algorithm can be used to compute the trail cover number without storing trails. It can be implemented to build the computation up from leaves and thus run in linear time and space.

The intuition behind the algorithm is that deleting a vertex from a minimum trail cover should leave minimum trail covers of the resulting subtrees. The code $c(T, x)$ computed in the algorithm takes care of the fact that this is not true for arbitrary minimum trail covers. This is illustrated by the tree T on the left in Figure 2.

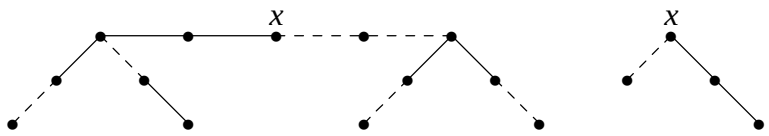

Figure 2. Insuffi cient variants of optimality

There are several ways to cover the edges of this tree with three trails; the Figure shows one of them in solid edges. Deleting x leaves two subtrees that can be covered with one trail each. The only minimum covering of T that turns into minimum coverings of the subtrees when x is deleted is the one in which x is a degenerate trail. In the algorithm, this corresponds to the case $k_1 = k_2 = k_3 = 0$. This example suggests that a minimum covering in which the root is a degenerate trail is desirable if one exists. We say that a minimum trail cover $\mathbf{C}$ of a tree T with root x is *weakly optimal* if $\mathbf{C}$ ends at x or no minimum cover ends at x. Furthermore, $\mathbf{C}$ is *strongly optimal* if (a) $\mathbf{C}$ is weakly optimal and (b) $\mathbf{C}$ isolates x or no minimum cover isolates x. The statements ‘‘no minimum cover ends at x’’ and ‘‘no minimum cover isolates x’’ are equivalent to $c(T, x) \le 1$ and $c(T, x) \le 2$, respectively. The empty trail cover is a strongly optimal trail cover of the 1-vertex tree. The next theorem is a precise version of the intuition suggested above and is the main result needed to prove the correctness of the algorithm.

THEOREM 3.2. If $\mathbf{C}$ is a strongly optimal trail cover of (T, x), then the trails in $T - x$ obtained by deleting x from any trail containing it in $\mathbf{C}$ form weakly optimal trail covers of the components of $T - x$ rooted at the neighbors of x.

Proof. Let $\{x_i\}$ be the neighbors of x, $\{T_i\}$ the subtrees, and $\{\mathbf{C}_i\}$ the resulting sets of trails. Suppose $\mathbf{C}_i$ is not weakly optimal, and let $\mathbf{D}_i$ be a weakly optimal trail cover of (T_i, x_i). In all cases, we construct a trail cover of (T, x) that contradicts the strong optimality of $\mathbf{C}$.

Case 1. **C** does not use the edge xx_i. In this case, all the trails of $\mathbf{C}_i$ are trails of **C**. If $|\mathbf{D}_i| < |\mathbf{C}_i|$, let $\mathbf{D}' = \mathbf{C} - \mathbf{C}_i + \mathbf{D}_i$. Since $|\mathbf{D}'| < |\mathbf{C}|$ and **C** is strongly optimal, $\mathbf{D}'$ is not a trail cover, so it fails to touch some edge. The only edge that $\mathbf{D}'$ can miss is xx_i; if $\mathbf{D}'$ misses xx_i, then **C** does not visit x. If this happens, then $\mathbf{D} = \mathbf{D}' \cup \{(x)\}$ is a trail cover with $|\mathbf{D}| \le |\mathbf{C}|$ that isolates x. This contradicts the strong optimality of **C**, because **C** does not visit x.

Hence we may assume $|\mathbf{D}_i| = |\mathbf{C}_i|$. Since we assumed $\mathbf{C}_i$ is not weakly optimal, the weak optimality of $\mathbf{D}_i$ implies that $\mathbf{D}_i$ ends at x_i and $\mathbf{C}_i$ does not. Let D be the trail in $\mathbf{D}_i$ that ends at x_i, and let $\mathbf{D}_i'$ be the collection of trails obtained from $\mathbf{D}_i$ by extending D to include the edge $x_i x$. Let $\mathbf{D} = \mathbf{C} - \mathbf{C}_i + \mathbf{D}_i'$; note that $|\mathbf{D}| = |\mathbf{C}|$. If **C** ends at x, then **D** has two trails ending at x that can be concatenated to obtain a smaller trail cover than **C**. If **C** does not end at x, then **D** is a trail cover of the same size that ends at x. Each possibility contradicts the strong optimality of **C**.

Case 2. **C** uses the edge xx_i on some trail C. In this case $\mathbf{C}_i$ ends at x_i. Since $\mathbf{C}_i$ is not weakly optimal, this implies $|\mathbf{D}_i| < |\mathbf{C}_i|$. Let **D** be the collection of trails obtained from **C** by deleting xx_i from C and replacing the resulting $\mathbf{C}_i$ by $\mathbf{D}_i$. Note that **D** ends at x, that $|\mathbf{D}| \le |\mathbf{C}|$, and that **D** is a trail cover of T.

If C ends at x, then **D** isolates x but **C** does not, which contradicts the strong optimality of **C**. If C does not end at x, then we consider two possibilities, as in the second half of Case 1. If **C** ends at x, then **D** has two trails ending at x that can be concatenated to obtain a smaller trail cover than **C** . If **C** does not end at x, then **D** is a trail cover of at most the same size that ends at x. Each possibility contradicts the strong optimality of **C**. ■

The trees in Figure 2 show that both types of optimality are necessary. The trail cover given by solid edges for the tree on the left shows that deletion of the root from a weakly optimal trail cover need not leave minimum trail covers for the subtrees. The trail cover indicated for the tree on the right shows that deletion of the root from a strongly optimal trail cover need not leave strongly optimal covers for the subtrees.

If **C** is a strongly optimal trail cover of (T, x), then applying the Algorithm to the resulting covers $\{\mathbf{C}_i\}$ of the subtrees reconstructs **C** or constructs another trail cover with the same number and placement of trail ends as **C**. To prove the correctness of the algorithm, we will show that applying the algorithm to arbitrary weakly optimal trail covers of the rooted subtrees generates a strongly optimal trail cover of (T, x).

LEMMA 3.3. If some weakly optimal trail cover **C** of a rooted tree (T, x) visits x but does not end at x, then every minimum trail cover of (T, x) is strongly optimal.

Proof. In this situation the definition of weakly optimal implies that $c(T, x) = 1$ and no minimum trail cover ends at x. Hence every weakly optimal trail cover is strongly optimal. A minimum trail cover $\mathbf{C}'$ can fail to be strongly optimal only if it does not visit x. Since x does not appear $\mathbf{C}'$ consists of $|\mathbf{C}'| = |\mathbf{C}|$ trails covering $T - x$. If we delete x from **C**, we obtain at least $|\mathbf{C}| + 1$ trails, since **C** does not end at x. This implies that at least one of the components of $T - x$ does not receive a minimum trail cover, which contradicts Theorem 3.2. ■

In the application of the recursive step, Algorithm 3.1 treats subtrees with code 2 or 3 exactly the same. Hence it will behave the same and produce the same code and size of trail cover for the root as long as the trail covers of the subtrees are any weakly optimal trail covers $\mathbf{C}_i$. Phrasing the recursive step of the algorithm in terms of the trail cover alone, it (1) forms the union of the $\mathbf{C}_i$, (2) extends to x any trail ending at a neighbor x_i, (3) concatenates pairs of trails extended to x until at most one remains, and (4) adds (x) as a trail of length 0 if no trail extended to x and some neighbor x_i is not visited by its weakly optimal $\mathbf{C}_i$. To complete the proof, it suffices to show that the resulting $\mathbf{C}$ is strongly optimal.

THEOREM 3.4. If (T, x) is a rooted tree with neighbors $\{x_i\}$ of the root, and $\{\mathbf{C}_i\}$ are weakly optimal trail covers of the rooted subtrees $\{(T_i, x_i)\}$ of $T - x$, then application of the recursive step of Algorithm 3.1 to $\{\mathbf{C}_i\}$ produces a strongly optimal trail cover $\mathbf{C}$ of (T, x).

Proof. Suppose that $\mathbf{D}$ is a strongly optimal trail cover of (T, x) and $\{\mathbf{D}_i\}$ are obtained from $\mathbf{D}$ by deleting all appearances of x. By Theorem 3.2, $\mathbf{D}_i$ is a weakly optimal trail cover of (T_i, x_i). By the definition of weak optimality, $\mathbf{D}_i$ ends at x_i if and only if $\mathbf{C}_i$ ends at x_i. Hence applying the algorithm to $\{\mathbf{C}_i\}$ produces the same number of extensions to x as applying the algorithm to $\{\mathbf{D}_i\}$ to obtain $\mathbf{D}$. As a result, $\mathbf{C}$ and $\mathbf{D}$ have the same size and associated code, except possibly when there are no extensions to x.

In this final case, for each i neither $\mathbf{D}_i$ nor $\mathbf{C}_i$ ends at x_i. We still have identical size and code for $\mathbf{C}$ and $\mathbf{D}$ unless one of them visits all $\{x_i\}$ and the other does not. This possibility is forbidden by Lemma 3.3. ■

4. CRITICAL TREES

To characterize the trees with interval number $m + t$, we characterize the trees that just barely require t trails in a trail cover. The graph obtained by contracting an edge e of G is denoted $G \cdot e$. An edge e is *contractible* if $t(G \cdot e) = t(G)$, and a tree is *critical* if it has no contractible edge. We seek to characterize the *k-critical* trees, which are the critical trees with trail cover number k.

By applying Algorithm 3.1 to (G, x) for arbitrary $x \in V(G)$, we obtain a unique code for any vertex of a tree G. We say that a tree vertex v is *essential* if $c(G, v) = 3$, *useful* if $c(G, v) = 2$, *not useful* if $c(G, v) < 2$, and *at least useful* if $c(G, v) \geq 2$. The terminology is chosen to suggest that the critical trees are those in which every vertex is at least useful. A *penultimate* vertex of a tree is a non-leaf that has at most one non-leaf neighbor. We refer to a vertex of degree 2 as a *bivalent* vertex and an edge incident to a leaf as a *pendant edge*.

LEMMA 4.1. Given a tree G, (1) every penultimate vertex is at least useful, (2) the non-leaf neighbor of a penultimate vertex is not essential, and (3) if G is critical, then every penultimate vertex is bivalent.

Proof. Let u be a penultimate vertex, and let v be its non-leaf neighbor. (1) We can modify any minimum trail cover so that it touches the pendant edges incident to u via a trail ending at u, by moving a degenerate trail from a leaf to u or deleting the leaf from a trail containing u. (2) If $\mathbf{C}$ is a minimum trail cover isolating v, then some other trail must touch the edges from u to its leaf neighbors. As above, we may assume this trail ends at u. Extending this trail to v produces a smaller trail cover. (3)

If u is not bivalent, let G' be the tree obtained by contracting all but one pendant edge incident to u. Since u is penultimate in G', G' has a minimum trail cover that ends at u. This is also a trail cover of G, so G was not critical. ■

Having introduced these elementary properties of critical trees, let us describe how to grow the trees we will show are the critical trees. Suppose that G is a tree with a vertex partition into two sets U and W. Let P be a 5-vertex path with central vertex u, partitioned so that $U = \{u\}$ and $W = V(P) - u$. Then the *augmentation of G at v* is the tree G' obtained from $G \cup P$ by (1) adding the edge uv if $v \in W$ or (2) identifying u, v if $v \in U$ (the combined vertex remains in U). Let $\mathbf{H}_1 = \{K_2\}$ with both vertices in W, and let $\mathbf{H}_k$ be the collection of all augmentations of trees in $\mathbf{H}_{k-1}$. The graphs in $\mathbf{H}_1$, $\mathbf{H}_2$, and $\mathbf{H}_3$ appear in Figure 3, with the vertices of U indicated by open dots and those of W by closed dots. The graph in $\mathbf{H}_2$ is the unique forbidden subtree for trees that are interval graphs [10]. The three graphs of $\mathbf{H}_3$ correspond to augmentations at the three isomorphism classes of vertices of the graph in $\mathbf{H}_2$.

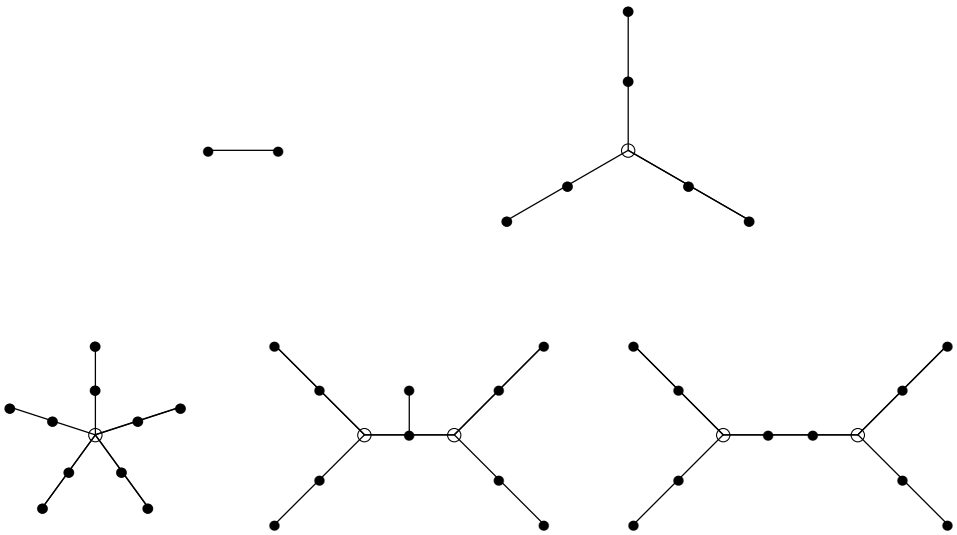

Figure 3. The sets $\mathbf{H}_1$, $\mathbf{H}_2$, and $\mathbf{H}_3$.

We need one more concept describing the relationship between vertices and trail covers: we say that a trail cover $\mathbf{C}$ *swallows* v if v is an internal vertex of some trail in $\mathbf{C}$. Note that a minimum trail cover may end at v and swallow v, but it cannot isolate v and swallow v.

THEOREM 4.2. If $G \in \mathbf{H}_k$, with $V(G) = U \cup W$ as described in the construction of $\mathbf{H}_k$, then G is k-critical, the set of essential vertices in G is W, and the set of useful vertices in G is U. Furthermore, every minimum trail cover of G swallows every vertex of U.

Proof. By induction on k. The claim holds by inspection for $k = 1$. Suppose that $k > 1$ and that G is the augmentation of $G' \in \mathbf{H}_{k-1}$ at v. We will apply Algorithm 3.1 to (G, u), where $u = v$ if $v \in U$ and u is the new neighbor of v (on P) if $v \in W$.

Case 1. $v \in W$. In this case, u has exactly three neighbors. The subtrees for the algorithm are (G', v) and two copies of K_2. By the induction hypothesis, v is essential in G', so $c(G', v) = 3$. The

code for any vertex of K_2 is also 3. The algorithm therefore yields $t(G) = 1 + t(G') = k$ and $c(G, u) = 2$. Hence $u \in U$, and trails in the subtrees can be extended or shifted in such a way that a degenerate trail is left at any other vertex of P to prove $V(P) - \{u\} \subset W$. Furthermore, if some minimum trail cover does not swallow u, then the pendant edges of P require two trails wholly contained in P, forcing an impossible trail cover of G' with size $k - 2$.

The induction hypothesis guarantees that vertices of U in G' are at least useful in G' and vertices of W in G' are essential in G'. A minimum trail cover establishing this for a particular vertex of G' can be combined with P to obtain a minimum trail cover establishing the same condition for this vertex in G. Similarly, for any edge e in G', we can combine a minimum trail cover of $G' \cdot e$ with P to show that e is not contractible in G. If we contract any edge of P, then the algorithm applied at u will use only $k - 1$ trails, since one of the subtrees becomes K_1. If we contract the edge uv, then we can replace the degenerate trail (v) by the trail P in some minimum trail cover of G' to obtain $k - 1$ disjoint trails covering $G \cdot uv$.

We have shown that G is k-critical; it remains only to show that a vertex $w \in U \cap V(G')$ is not essential in G and is swallowed by any minimum trail cover. By the induction hypothesis, any trail cover $\mathbf{C}'$ of G' that isolates w or does not swallow w has at least k trails. If $\mathbf{C}$ is a minimum trail cover of G that isolates w or does not swallow w, then restricting these trails to G' creates such a trail cover $\mathbf{C}'$ of G'. However, the pendant edges of P require $\mathbf{C}$ to have a trail that contains no vertex of G'. Hence $|\mathbf{C}| \geq k + 1$, and the assumption on $\mathbf{C}$ and w was impossible.

Case 2. $v \in U$. In this case $u = v$. The subtrees for applying the algorithm to (G, u) are the same as for applying it to (G', v), plus two copies of K_2. If β is the value of $k_2 + k_3$ for (G, u) and β' is its value for (G', u), then $\beta = \beta' + 2$, since vertices of K_2 have code 3. By the induction hypothesis, v is useful in G', so $c(G', v) = 2$. The algorithm therefore yields $t(G) = 1 + t(G') = k$ and $c(G, u) = 2$. Hence $u \in U$, and again trails in the subtrees can be extended or shifted in such a way that a degenerate trail is left at any other vertex of P to prove $V(P) - \{u\} \subset W$. Furthermore, if some minimum trail cover does not swallow u, then the pendant edges of P require two trails wholly contained in P. This leaves a trail cover of G' of size $k - 2$ unless one of the trails in P ends at u, in which case we have a trail cover of G' of size $k - 1$ that isolates v; both cases are forbidden by the induction hypothesis.

The remainder of the proof for this case is identical to the last two paragraphs of the proof for Case 1, except that the last sentence of the first paragraph (on contracting the edge uv) is unnecessary and should be deleted, the second paragraph applies only to $w \in (U \cap V(G') - \{v\}$, and the last two sentences of the second paragraph should be replaced by the following: ''However, the pendant edges of P require $\mathbf{C}$ to have a trail that contains no vertex of G' except possibly v. This forces $|\mathbf{C}| > |\mathbf{C}'|$ unless v is a degenerate trail in $|\mathbf{C}'|$, in which case $|\mathbf{C}'| \geq k$ because $c(G', v) = 2$. In either case we have $|\mathbf{C}| > k$, and the assumption on $\mathbf{C}$ and w was impossible.'' ■

To complete the characterization of the critical trees, we need only show that every critical tree arises in this way.

THEOREM 4.3. For each $k \geq 1$, the set of k-critical trees is $\mathbf{H}_k$.

Proof. We need only show that every k-critical tree is in $\mathbf{H}_k$; we use induction on k. The claim is immediate for $k = 1$; suppose $k > 1$ and G is k-critical. Let P be a longest path in G. Since penultimate vertices in critical trees are bivalent, a critical tree with longest path having fewer than five

vertices is only a path. Hence we may assume $P = (u, v, w, x, \ldots, z)$. We have $d(v) = 2$. If $d(w) = 2$, then uv is contractible. If w is incident to a pendant edge e (not in P), then e is contractible. Hence $d(w) \geq 3$ and every neighbor of w except (possibly) x is penultimate (since P is a longest path).

Let $r = d(w) - 1$, let $v_1, \ldots, v_r$ be the neighbors of w other than x, and let $u_1, \ldots, u_r$ be the leaves adjacent to them. When using Algorithm 3.1 on (G, x), we build a minimum covering of the subtree rooted at w by concatenating trails from the v_i in pairs. In particular, the path $Q = (v_1, w, v_2)$ is a trail in some minimum covering $\mathbf{C}$ of G. Let $G' = G - \{u_1, v_1, u_2, v_2\}$ if $r > 2$, and let $G' = G - \{u_1, v_1, u_2, v_2, w\}$ if $r = 2$. Since Q does not extend to x when the algorithm is applied to x, $\mathbf{C} - \{Q\}$ is a minimum covering of G'. The k-criticality of G then implies that G' is $(k-1)$-critical, which by the induction hypothesis implies $G' \in \mathbf{H}_{k-1}$. To show that G is an augmentation of G' and thus that $G \in \mathbf{H}_k$, we consider three cases for the value of r.

First, it cannot happen that r is odd. In this case, the algorithm for (G, x) builds a trail rising from w to x. In $(G' \cdot wx, x')$, where x' denotes the combined vertex, the same trail results, with the edge wx contracted. Hence the trail cover number computed by the algorithm is the same for G and $G \cdot wx$, contradicting the criticality of G.

If $r = 2$, we show that $x \in V(G')$ is in W in the canonical partition of G', and hence G is the augmentation of G' at x. If $x \in U$, then by Theorem 4.2, G' has no trail cover of size $k-1$ that isolates x. This means that the trail ending at x in a minimum trail cover of G' cannot be extended to touch both u_1v_1 and u_2v_2 in $G \cdot wx$. As a result, $t(G \cdot wx) = k$ and wx is contractible, contradicting the criticality of G.

If r is even and $r > 2$, then w has penultimate neighbors in G'. By Lemma 4.1, w cannot be essential in G'. By Theorem 4.2, this implies $w \in U$ in the canonical partition of G', and hence G is the augmentation of G' at w. ■

5. APPLICATIONS

We close this paper by using our results for trees to give alternative proofs of two results that appeared earlier. These are the maximum value of $I(G)$ when G is an n-vertex tree (Aigner-Andreae [1]) and the maximum value of $I(G)$ when G is a connected graph with m edges (Kratzke-West [7]). We use the notation and concepts of the earlier sections, together with the auxiliary function $f(G) = n(G) - 4t(G) + 1$.

COROLLARY 5.1. If G is a tree with $n \geq 3$ vertices, then $I(G) \leq (5n - 3)/4$.

Proof. It suffices to show that $f(G) \geq 0$ for every tree G. If not, let G be a smallest tree such that $f(G) < 0$. If the contraction of some edge does not decrease the trail cover number, then it decreases f by one. Hence G must be critical, which implies $G \in \cup \mathbf{H}_k$. Each such graph is obtained by a sequence of augmentations from K_2. An augmentation at a vertex of U increases f by one, and an augmentation at a vertex of W leaves f unchanged. Although $f(K_2) = -1$, K_2 has no vertex in W, so the first augmentation changes f to 0, and thereafter $f \geq 0$ for each graph in $\cup \mathbf{H}_k$. ■

A closer look at this yields a "procedure" for computing f and a description of the n-vertex trees with maximum total interval number, equal to $\lfloor (5n - 3)/4 \rfloor$. Aigner and Andreae [1] presented

trees achieving the bound. A tree G achieves this value if and only if $f(G) \le 3$. For $G \in \mathbf{H}_k$, the size of U in the canonical partition of G is the number of augmentations at black vertices in the construction of G from K_2. Hence we can determine $f(G)$ for an arbitrary tree G as follows: (0) Initialize $f = -1$. (1) While the remaining graph has a contractible edge, contract it and increase f by 1. (2) When the remaining graph has no contractible edge, determine its canonical partition, increase f by $|U|$, and terminate.

Reworded, this means that the trees with $f(G) = k$ are (1) the trees having a contractible edge e such that $f(G \cdot e) = k - 1$ and (2) the critical trees having a canonical partition U, W with $k + 1$ vertices in U. Using this and the fact that K_2 is the only tree with $f(G) = -1$, it is not hard to describe explicitly the trees with $f \le 1$. The trees $\mathbf{F}$ with $f = 0$ are obtained by subdividing every edge of a star that has an odd number of edges (the first of these is P_3 and is not critical). The trees with $f = 1$ consist of those obtained by adding a pendant (contractible) edge to a tree with $f = 0$ and those in the three families in Figure 4, where the dashed central edge on the left is contractible and the open circles indicate vertices of U.

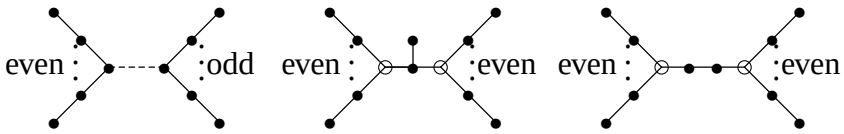

Figure 4. Trees with $f = 1$.

In [7], we proved by ad hoc inductive methods that $I(G) \le (5m + 2)/4$ for every connected graph with m edges. Our results for trees yield the bound more cleanly. Since $n(G) = m(G) + 1$ when G is a tree, we have $I(G) = (5m + 2)/4$ when G is a tree with $f(G) = 0$. Here we prove that any other connected graph with at least two edges has smaller total interval number. In fact, we bound the depth-2 total interval number, again by using trail covers.

THEOREM 5.2. The extremal trees are the unique connected graphs having the maximum total interval number in terms of size. In particular, if G is a connected graph with $m \ge 2$ edges and $G \notin \mathbf{F}$, then $I(G) \le I_2(G) < (5m + 2)/4$.

Proof. We use induction on the number of cycles in G. If G is a tree, then the result follows from the characterization of the trees with $f = 0$. If G has a cycle, we alter G in a way that reduces the number of cycles without changing the number of edges or reducing the trail cover number.

Given distinct edges uv and vw in G, define *snipping* uv to mean deleting uv and subdividing vw. To see that snipping uv does not reduce the trail cover number, let x be the new vertex in the resulting graph G'. A trail cover of G' can be turned into a trail cover of G with the same size by contracting vx.

If G has an edge e that belongs to a cycle of G but not to every cycle of G, then snipping e leaves a graph G' that still has a cycle, and the induction hypothesis yields the result. Otherwise, G has exactly one cycle, and snipping any edge of this cycle yields a tree G'. If $f(G') > 0$, then again we are finished. Hence we may assume that every snip of any edge on the unique cycle in G yields the unique m-edge tree G' with $f(G') = 0$. However, this is impossible; if snipping by deleting uv and subdividing vw yields the extremal graph, then snipping by deleting vw and subdividing uv does not. ■

References

[1] M. Aigner and T. Andreae, The total interval number of a graph. *J. Comb. Theory (B)* (1988).

[2] A.A. Bertossi, The edge Hamiltonian path problem is NP-complete. *Info. Proc. Letters* 13(1981), 157-159.

[3] M.R. Garey, D.S. Johnson, and R.E. Tarjan, The planar Hamiltonian circuit problem is NP-complete. *SIAM J. on Computing* 5(1976), 704-714.

[4] J.R. Griggs and D.B. West, Extremal values of the interval number of a graph, I. *SIAM J. Algebraic and Discrete Methods* 1(1980), 1-7.

[5] F. Harary and C.St.J.A. Nash-Williams, On eulerian and hamiltonian graphs and line graphs. *Canad. Math. Bull.* 8(1965), 701-710.

[6] T.M. Kratzke, The total interval number of a graph. Ph.D. Thesis, Univ. of Illinois (1987), Coordinated Science Laboratory Research Report UILU-ENG-88-2202.

[7] T.M. Kratzke and D.B. West, The total interval number of a graph, I: Fundamental classes. *Discrete Math.* 118(1993), 145-156.

[8] P.G.H. Lehot, An optimal algorithm to detect a line-graph and output its root graph, *J. ACM* 21(1974), 569-575.

[9] D.B. Shmoys and D.B. West, Recognizing graphs with fixed interval number is NP-complete. *Discrete Appl. Math.* 8(1984), 295-305.

[10] W.T. Trotter and F. Harary, On double and multiple interval graphs. *J. Graph Theory* 2(1978), 137-142.