# Generalized centrality in trees

Henry Martyn Mulder
Econometrisch Instituut, Erasmus Universiteit,
P.O. Box 1738, 3000 DR Rotterdam, The Netherlands
hmmulder@few.eur.nl

Michael J. Pelsmajer
Illinois Institute of Technology, Department of Applied Mathematics
Chicago, IL 60616
pelsmajer@iit.edu

K. B. Reid
Department of Mathematics
California State University San Marcos
San Marcos, California 92096-0001
breid@csusm.edu

April 18, 2006

Econometric Institute Report EI 2006-16

**Abstract**

In 1982, Slater defined path subgraph analogues to the center, median, and (branch or branchweight) centroid of a tree. We define three families of central substructures of trees, including three types of central subtrees of degree at most $D$ that yield the center, median, and centroid for $D = 0$ and Slater's path analogues for $D = 2$. We generalize these results concerning paths and include proofs that each type of generalized center and generalized centroid is unique. We also present algorithms for finding one or all generalized central substructures of each type.

## 1 Introduction

For many purposes one is interested in determining the "middle" of the graph. For instance, already in 1869 Jordan [11] used this idea in the case of trees to determine the automorphism group of a tree. From the viewpoint of applications, an interesting example is the placing of a facility on a network: given a set of clients that has to be serviced by the facility, one wants to find a location for the facility that optimizes certain criteria. It turns out that even in the case of trees, there is no such uniquely determined "middle" of a tree; it very much depends on the problem at hand. For example, if the facility is a fire station, then one wants to minimize the maximum distance to the flammable objects. Whereas if the facility is a distribution center for a set of warehouses, then the sum of the distances to all the warehouses is a determining factor in the cost of servicing the warehouses. To date, there is an abundance of all kinds of centrality notions for trees, and many (but not all) have natural generalizations for arbitrary graphs. The two classical examples of Jordan [11] are the *center* of a tree (the set of vertices that minimize the maximum distance to other vertices) and the *centroid* of a tree (the set of vertices $x$ in the tree $T$ that minimize the maximum order of a component of $T - x$.) A third type of middle is the *median* (the set of vertices that minimize the sum of the distances—or, equivalently, the average distance—to other vertices). The center and the median have natural generalizations for arbitrary graphs, while it is less obvious how to generalize the centroid to arbitrary graphs [17].

Slater took the first step in generalizing these notions, although in a different sense [21]: he considered paths in a tree that minimize the appropriate criteria. In this case, the notions of center, centroid, and median lead to three different optimal paths: *central path*, *spine*, and *core*, respectively, in the terminology of [21].

A path in a tree is a subtree of maximum degree at most 2. So a natural next step is to consider subtrees of maximum degree at most $D$. Thus we get the *$\mathcal{T}_D$-center*, *$\mathcal{T}_D$-centroid*, and *$\mathcal{T}_D$-median*. More importantly, we extend these ideas to quite general families of subtrees that include the family of paths and families of trees of maximum degree at most $D$. Thus our definitions generalize centers, centroids, medians, central paths, spines, and cores, and in fact our results generalize known results for these six as well. It also generalizes work by McMorris and Reid [14] (following Minnieka [15]) on subtrees of order $k$ that minimize eccentricity. Our work also includes the notion of a central caterpillar, suggested by McMorris [13]. While there have been other many other generalizations of the center and centroid (some of which we discuss in Section 5), none directly generalize central paths or spines. There have been some very interesting generalizations of a core [19, 20, 23, 24, 1]. In particular, the *$k$-tree core* is similar to our $\mathcal{T}_D$-median, except that instead of bounding the maximum degree, this concept considers subtrees with at most $k$ leaves. There are linear time algorithms for finding a $k$-tree core of a tree when $k$ is fixed [19, 20].

We will give linear time algorithms for finding the unique $T_D$-center, the unique $T_D$-centroid, a $\mathcal{T}_D$-median, and the set of vertices contained in $\mathcal{T}_D$-medians, for any tree when $D$ is fixed. We will also show how to obtain all of the $\mathcal{T}_D$-medians in time that is linear in the sum of the order of $G$ and the number of $\mathcal{T}_D$-medians. Also, every one of our generalizations of a center and centroid of a tree $T$ gives a unique member of a small family of subtrees of $T$ that can be produced in linear time.

## 2 Preliminaries

Let $G = (V, E)$ be a connected graph with vertex set $V$ and edge set $E$. For any subgraph $H$ of $G$, we denote its vertex set by $V(H)$. The subgraph of $G$ induced by a subset $W$ of $V$ is denoted by $G[W]$. The *order* of a (sub)graph is the number of vertices in the (sub)graph. The *length* $l(P)$ of a path $P$ in $G$ is the number of edges in $P$. The *distance* $d(u, v)$ between vertices $u$ and $v$ in $G$ is the length of a shortest path between $u$ and $v$. The *eccentricity* $e(v)$ of a vertex $v$ in $G$ is the maximum of the distances from $v$ to the other vertices in $G$. The *center* $C(G)$ of $G$ is the set of vertices of $G$ with minimum eccentricity. The *status* $s(v)$ of a vertex $v$ is the sum of the distances from $v$ to all other vertices in $G$. Clearly, $\frac{s(v)}{|V|-1}$ is the average distance from $v$ to the other vertices of $G$. A *median vertex* is a vertex that minimizes the status. The *median* $M(G)$ of $G$ is the set of median vertices of $G$.

In the case that $G$ is a tree, the *branchweight* $bw(v)$ of a vertex $v$ in $G$ is the order (i.e., number of vertices) of the largest component of $G - v$. The *centroid* $B(G)$ of $G$ is the set of all vertices with minimum branchweight. For a tree, the centroid and median are identical [25], while the median can be arbitrarily far apart from the center. (In fact, given any two graphs $H_1$ and $H_2$ and any positive integer $k$, there exists a connected graph $H$ such that $H_1$ and $H_2$ are induced subgraphs of $G$ with $C(H) = V(H_1)$, $M(H) = V(H_2)$, and the distance between $H_1$ and $H_2$ is $k$, where the distance between two subgraphs is the minimum of all possible distances between a vertex from one subgraph to a vertex of the other [10]). In the classical paper by Jordan [11] it was already proved that $C(G)$ and $M(G)$ each consist of one vertex or two adjacent vertices. Linear time algorithms for finding each can be found in [3, 5, 6, 9].

Slater [21] generalized each of these notions to path subgraphs. If $X$ is a subgraph of $G$ or a subset of $V$, then the distance $d(X, v)$ from $X$ to a vertex $v$ in $G$ is the minimum of the distances from vertices of $X$ to $v$. Note that $d(X, u) = 0$ for any vertex $u$ in $X$. The *eccentricity* $e(X)$ of $X$ is the maximum of the distances from $X$ to the vertices of $G$. A *path center* (or *central path*) of $G$ is a path of minimum length among the paths in $G$ of minimum eccentricity. The *status* $s(X)$ of $X$ is the sum of the distances from $X$ to all other vertices of $G$. A *path median* (or *core*) is a path $P$ of $G$ that minimizes $s(P)$. In case $G$ is a tree, the *branchweight* $bw(X)$ of $X$ is the the order of the largest component of $G - V(X)$ ($G - X$ when $X$ is a set of vertices). A *path centroid* (or *spine*) is a path of minimum length among the paths $P$ of $G$ that minimizes $bw(P)$. It is possible for a path center, a spine, and a core to be distinct [21]. The path center and spine are unique for trees, contain $C(G)$ and $M(G)$ respectively, and there are simple linear time algorithms for computing each [21]. However, there may be many cores of a tree, a core need not contain $M(G)$, and while there are linear time algorithms for finding a core of a tree and the set of vertices that are contained in cores [16], we will see that there may be a superpolynomial number of cores.

In this paper, we generalize these central paths of a tree to central subtrees of a tree.

**Definition 2.1** Fix a tree $G$ and let $\mathcal{T}_D$ be the set of subtrees of $G$ with maximum degree at most $D$.

A *$\mathcal{T}_D$-center* of $G$ is a tree of least order in the set $\{T \in \mathcal{T}_D \colon e(T) \leq e(T') \ \forall\ T' \in \mathcal{T}_D\}$.

A $\mathcal{T}_D$-*centroid* of $G$ is a tree of least order in the set $\{T \in \mathcal{T}_D\colon bw(T) \le bw(T') \ \forall\ T' \in \mathcal{T}_D\}$.

A $\mathcal{T}_D$-*median* of a tree $G$ is a tree in the set $\{T \in \mathcal{T}_D\colon s(T) \le s(T') \ \forall\ T' \in \mathcal{T}_D\}$.

Note that for $D = 0$ these definitions yield the vertices of the center, centroid, and median of $G$, respectively (and is not unique if these sets contain two vertices). The $\mathcal{T}_1$-center and $\mathcal{T}_1$-centroid are the subtrees induced by the center and centroid of $G$, respectively. If there are two vertices in the median or if $|V(G)| = 1$, then the $\mathcal{T}_1$-median is the subtree induced by the median. Otherwise, each $\mathcal{T}_1$-median is induced by the unique median vertex $x$ and one neighbor of $x$ in a component of $G - x$ of largest order. Since paths are the trees of maximum degree 2, when $D = 2$ the above definitions become the path center, path centroid, and path median of a tree. When $D$ is greater than or equal to the maximum degree of $G$, each of the definitions simply yields $G$.

## 3 Generalized path centers and path centroids

The following lemma is key to understanding our generalizations of the path center, including the $\mathcal{T}_D$-center. Note that a subtree $T$ of a tree $G$ is *minimal* with respect to some property $P$ means that no proper subtree of $T$ has property $P$.

**Lemma 3.1** *If $G$ is a tree and $0 \le k \le e(C(G))$, then $G$ has a unique minimal subtree with eccentricity at most $k$.*

Note that if $C(G) = \{x\}$ then $e(C(G)) = e(x)$, but if $C(G) = \{x, y\}$ then $e(C(G)) = e(x) - 1 = e(y) - 1$.

One consequence of this lemma is that for a given tree $G$, if $T^*$ is the unique minimal subtree of $G$ with $e(T^*) \le k$, and $T$ is any subtree of $G$ with $e(T) \le k$, then $T^*$ is a subtree of $T$. This follows since a smallest subtree of $T$ with eccentricity less than or equal to $k$ is clearly minimal with respect to this property, so it must be equal to $T^*$.

**Proof.** We will induct on $e(C(G)) - k$ to find $X_k \subset V(G)$ such that $G[X_k]$ is the unique minimal subtree of $G$ with $e(G[X_k]) \le k$.

$G[C(G)]$ is a subtree of $G$ with eccentricity $e(C(G))$. It is easy to see that if $T$ is a subtree of $G$ that does not contain $C(G)$, then $e(T) > e(C(G))$. Therefore $G[C(G)]$ is the unique minimal subtree of $G$ with eccentricity at most $e(C(G))$, and $X_{e(C(G))} = C(G)$.

Suppose that $G[X_k]$ is the unique minimal subtree of $G$ with eccentricity at most $k$. If $T$ is a subtree of $G$ with eccentricity at most $k - 1$, then clearly $T$ has eccentricity at most $k$, so (by the remarks preceding this proof) $T$ must contain $G[X_k]$.

For each component $H$ of $G - X_k$, let $x_H y_H$ be the edge with $x_H \in X_k$ and $y_H \in V(H)$, and let $l(H)$ be the eccentricity of $x_H$ in $G[V(H) \cup \{x_H\}]$. Let $X_{k-1} = X_k \cup \{y_H\colon l(H) = k\}$. Note that $e(G[X_{k-1}]) = k - 1$, and that the eccentricity is larger for any subtree that does not intersect every $H$ for which $l(H) = k$.

Therefore $G[X_{k-1}]$ is the unique minimal subtree of $G$ with eccentricity at most $k - 1$. ■

Throughout this section, we will continue to use $X_k$ to represent the vertex sets generated in the proof of Lemma 3.1.

**Remark 3.2** The unique minimal subtrees in Lemma 3.1 are nested. That is, if $G$ is a tree and for $0 \leq k \leq e(C(G))$, $G[X_k]$ is unique minimal subtree of $G$ with eccentricity at most $k$, then $G[X_{e(C(G))}] \subseteq \ldots \subseteq G[X_1] \subseteq G[X_0]$. (Of course, $G[X_0] = G$.)

We will present a theorem that applies to a much more general class of central substructures than merely $\mathcal{T}_D$-centers. For this we introduce the following definitions.

**Definition 3.3** $\mathcal{T}$ is a *hereditary class of trees* if $\mathcal{T}$ is a nonempty set of trees such that for each $T \in \mathcal{T}$, every subtree of $T$ is in $\mathcal{T}$.

Observe that this resembles the definition of a hereditary class of graphs except that "set of trees" and "subtree" replaces "set of graphs" and "subgraph". However, it is not a special type of hereditary class of graphs because the subtrees of a tree are its *connected* subgraphs.

We first note a few easily checked facts.

**Proposition 3.4** *Let $\mathcal{T}$ be a hereditary class of trees. Then*

1. $K_1 \in \mathcal{T}$,
2. $K_2 \in \mathcal{T}$ *unless* $\mathcal{T} = \{K_1\}$,
3. *all subtrees of a fixed tree form a hereditary class of trees,*
4. *unions and intersections of hereditary classes of trees yield new hereditary classes of trees, and*
5. *if $\mathcal{T}$ is finite, then $\mathcal{T} = \bigcup\{\text{subtrees of } T\}$, where the union is taken over all $T$ such that $T$ is a maximal tree in $\mathcal{T}$.*

Many other observations can be easily generated. For example: If $|\mathcal{T}| \geq 2$, then $K_2 \in \mathcal{T}$. If $|\mathcal{T}| \geq 3$, then $T$ also contains $K_{1,2}$. If $|\mathcal{T}| = 4$, then $\mathcal{T}$ contains $K_1, K_2, K_{1,2}$, and either $K_{1,3}$ or $P_4$ (the path of order 4).

**Definition 3.5** Let $\mathcal{T}$ be a hereditary class of trees, let $G$ be a graph, and let $\mathcal{T}'$ be the subtrees of $G$ that are in $\mathcal{T}$. A $\mathcal{T}$*-center* of $G$ is an element of smallest order in the set $\{\, T \in \mathcal{T} : \; e(T) \leq e(T') \; \forall T' \in \mathcal{T}'\}$.

Note that this directly generalizes the $\mathcal{T}_D$-center from Definition 2.1, since $\mathcal{T}_D$ is the family of trees of maximum degree at most $D$, and $\mathcal{T}_D$ is clearly a hereditary class of trees.

Other examples include trees of order at most $k$, trees of diameter at most $d$ (for $d = 2$ these are stars), caterpillars (including all paths), lobsters (a lobster is a tree that contains a path of eccentricity at most 2), subdivisions of stars, and all the subtrees of any fixed set of trees. In addition, the union and intersection of two hereditary classes of trees are both hereditary classes of trees. The following theorem applies to each of these classes.

**Theorem 3.6** *For a tree $G$ and a hereditary class of trees $\mathcal{T}$, the $\mathcal{T}$-center of $G$ is unique unless both $\mathcal{T} = \{K_1\}$ and $|C(G)| = 2$.*

**Proof.** Clearly there is a $\mathcal{T}$-center of $G$. Let $T$ be one such subtree. If $e(C(G)) < e(T)$, then the subtree induced by $C(G)$ is not in $\mathcal{T}$. Then by Proposition 3.4(1), $|C(G)| \neq 1$. Hence $|C(G)| = 2$, and by Proposition 3.4(2), $\mathcal{T}$ must be $\{K_1\}$.

Otherwise we may apply Lemma 3.1, so $G$ has a unique minimal subtree $T^*$ with eccentricity at most $e(T)$. Every subtree of $G$ with eccentricity at most $e(T)$ contains $T^*$ as a subgraph; in particular, $T^* \subseteq T$. Since $\mathcal{T}$ is a hereditary class of trees, $T^* \in \mathcal{T}$. Since $T$ is a $\mathcal{T}$-center, its order is not greater than the order of $T^*$. Hence it must be that $T = T^*$. ■

**Corollary 3.7** *Let $G$ be a tree, and let $D$ be a positive integer. The $\mathcal{T}_D$-center is unique and contains the $T_{D'}$-centers of $G$ for all $1 \leq D' \leq D$.*

**Proof.** Theorem 3.6 with $\mathcal{T} = \mathcal{T}_D$ immediately implies that the $\mathcal{T}_D$-center $T$ is unique. Similarly, the $\mathcal{T}_{D'}$-center $T'$ is unique. Since $\mathcal{T}_{D'} \subseteq \mathcal{T}_D$, we have $e(T) \leq e(T')$ by definition. The proof of Theorem 3.6 shows that $T'$ is the unique minimal subtree with eccentricity at most $e(T')$. Then every subtree of $G$ with eccentricity at most $e(T')$ contains $T'$ as a subgraph; in particular, $T$ contains $T'$. ■

Similar results can be obtained for other hereditary classes of trees. For example, let $\mathcal{C}$ denote the set of caterpillars (including all paths), let $G$ be a tree, and let $\mathcal{C}'$ be the subtrees of $G$ that are in $\mathcal{C}$. A *caterpillar center* of $G$ is an element of smallest order in $\{T \in \mathcal{C}'\colon e(T) \leq e(T') \ \forall\ T' \in \mathcal{C}'\}$. Since $\mathcal{C}$ is a hereditary class of trees, each tree $G$ has a unique caterpillar center. Of course, if $G$ is a caterpillar then the caterpillar center of $G$ is $G$ itself. Note that the caterpillar center of a tree $G$ is usually not a path: The proof of Theorem 3.6 shows that the caterpillar center is $G[X_k]$, for some $k$. Now, if $k > 0$ and $G[X_k]$ is a path, then, according to the proof of Lemma 3.1, $G[X_{k-1}]$ is a path or a caterpillar and has eccentricity less than that of $G[X_k]$. This would contradict the fact that $G[X_k]$, being the caterpillar center, has smallest eccentricity among all caterpillars of $G$. The argument fails if $k = 0$ and $G[X_k]$ is a path, in which case $G = G[X_k]$. Hence the caterpillar center of $G$ is not a path unless $G$ is a path.

Similar results hold for other hereditary classes of trees, such as those mentioned above following Definition 3.5. Moreover, recall that McMorris and Reid [14] defined a central $k$-tree to be a subtree of order $k$ that minimizes eccentricity. A tree need not have a unique central $k$-tree. While subtrees of order $k$ do not form a hereditary class of subtrees, let $\mathcal{T}$ be the hereditary class of trees of order at most $k$. Then following McMorris and Reid [14], we see that their central $k$-trees of a tree are obtained from the $\mathcal{T}$-centers by adding arbitrary adjacent vertices until trees of order exactly $k$ are obtained.

We can find the $\mathcal{T}$-center quickly by finding each $X_k$ as $k$ decreases from $e(C(G))$ to 0, and stopping when $G[X_k]$ is not in $\mathcal{T}$. Then $G[X_{k+1}]$ is the $\mathcal{T}$-center. We wish to find $X_k$ quickly and test quickly whether $G[X_k] \in \mathcal{T}$.

**Theorem 3.8** *Let $G$ be a tree and let $G[X_k]$ be its unique minimal subtree with eccentricity at most $k$, for $0 \leq k \leq e(C(G))$. There is a linear time algorithm that finds all $X_k$ for $0 \leq k \leq e(C(G))$.*

**Proof.** First, $C(G)$ can be found in linear time (successively remove all leaves until $K_1$ or $K_2$ remains). If $C(G) = \{x\}$, we let $x$ be the root of $G$, and if $C(G) = \{x, y\}$ we contract $y$ to $x$ and let $x$ be the root of the "adjusted" tree (and we still refer to the adjusted tree as $G$). In the resulting rooted tree each non-root vertex $v$ is itself the root of a unique maximal tree $G_v$ induced by all vertices reachable from $x$ *via* $v$. Let $e'(v)$ be the eccentricity of $v$ in $G_v$. We determine $e'(v)$ for each $v$ in $V$ by a depth-first search (DFS) from the root $x$ as follows. Begin with $e'(v)$ set to 0 for all $v$ in $V$. When we arrive at a vertex $v$ from its child $u$, update $e'(v)$ to be the maximum of $e'(u) + 1$ and the current value of $e'(v)$. Observe that when the DFS is done, all $e'(v)$ are correctly computed in linear time. Note that $e'(x) = e(C(G))$. For each $0 \leq k \leq e(C(G))$ we create a list, and place a vertex $v \in V$ in the list with $k = e'(v)$ (in linear time). Now let $X_{e(C(G))} = C(G)$, and for each $k < e(C(G))$, let $X_k = X_{k+1} \cup \{v\colon e'(v) = k\}$. This can be done in linear time using the lists we set up. Thus, all $X_k$ are found in linear time, as desired. ■

It may be interesting to note that the set $X_k$ can also be generated as follows: Let $i = 0$. Repeatedly: let $Y_i$ be the set of current vertices of degree 1 (or 0), delete $Y_i$ while creating $Y_{i+1}$, and increment $i$. Continue until the tree is gone. Afterwards, $Y_k = \{v\colon e'(v) = k\}$, so let $X_k = X_{k+1} \cup Y_k$. This way can be implemented in linear time as well.

**Corollary 3.9** *Let $G$ be a tree and let $D$ be a positive integer. There is a linear time algorithm for finding the $\mathcal{T}_D$-center of $G$. There are similar linear time algorithms for finding the $\mathcal{T}$-center if $\mathcal{T}$ is the family of all of any of the following: any finite hereditary class of trees (such as trees of order at most $n$), stars (including $K_1$ and $K_2$), spiders (subdivisions of stars, including all stars), trees of diameter at most $D$, caterpillars (and paths), and lobsters (and caterpillars and paths).*

**Proof.** To find the $\mathcal{T}_D$-center in linear time, we modify the last step of Theorem 3.8 so that each time we add a vertex of $X_k - X_{k+1}$ to $X_k$, we check to see whether a vertex of degree greater than $D$ has been created in $G[X_k]$. When that happens, we stop and $G[X_{k+1}]$ is the $\mathcal{T}_D$-center. When $\mathcal{T}$ is the family of trees of order at most $n$, the $\mathcal{T}$-center can be found in linear time (much like the algorithm in [14]), since it is trivial to recognize the minimum $n$ such that $|X_{k+1}| > n$. (If $n = 1$ then $C(G)$ is the set of $\mathcal{T}$-centers, which can also be found in linear time.) Moreover, since it takes constant time to check whether a tree is isomorphic to a given fixed tree, whenever $\mathcal{T}$ is any a finite hereditary class of trees, the $\mathcal{T}$-center can be found in linear time. (As above, $\mathcal{T} = \{K_1\}$ is a special case.) If $\mathcal{T}$ represents stars (or spiders), one simply chooses the minimum $k$ such that $G[X_{k+1}]$ has more than one vertex of degree greater than 1 (degree greater than 2 for spiders). Thus the $\mathcal{T}$-center is found in linear time. Note that the diameter of $G[X_k]$ is 2 plus the diameter of $G[X_{k+1}]$ if $k \geq 0$, and the diameter of $G[X_{e(C(G))}]$ is 0 or 1 when $|C(G)|$ is 1 or 2, respectively. Then it is easy to see that if $\mathcal{T}$ is the family of trees of diameter at most $D$ (with $D \geq 1$), then $G[X_k]$ is the $\mathcal{T}$-center for $k = \max\{0, e(C(G)) - \lceil (D - |C(G)|)/2 \rceil\}$.

To find the central caterpillar in linear time, we begin by finding $k$ such that $G[X_k]$ is the $\mathcal{T}_2$-center (i.e., path center), as above. Note that this is not so different from the algorithm for finding a path center given in [21]. Unless $G$ is itself a path, $G[X_{k-1}]$ is not a path, in which case it is a caterpillar. Note that no vertex of $X_{k-1} - X_k$ is a leaf in

$G[X_{k-2}]$ unless $G[X_{k-1}] = G$ (and $k = 1$). Therefore, $G[X_{k-2}]$ is not a caterpillar when $k \geq 2$, so $G[X_{k-1}]$ is the central caterpillar unless $G$ is a path. Similarly, $G[X_{k-2}]$ is the lobster center unless $G$ is a caterpillar or a path, and thus the lobster center can be found in linear time. ■

For an arbitrary hereditary class of trees $\mathcal{T}$, the running time depends on how easy it is to recognize whether each $G[X_k]$ is in $\mathcal{T}$.

Similar results follow when the branchweight function $bw(\cdot)$ is used in place of eccentricity $e(\cdot)$. In particular, the next three results have nearly identical proofs to their eccentricity analogues, so we omit the proofs.

**Lemma 3.10** *If $G$ is a tree and $0 \leq k \leq bw(M(G))$, then $G$ has a unique minimal subtree with branchweight at most $k$.*

**Definition 3.11** Let $\mathcal{T}$ be a hereditary class of trees, let $G$ be a graph, and let $\mathcal{T}_G$ be the subtrees of $G$ that are in $\mathcal{T}$. A *$\mathcal{T}$-centroid of $G$* is an element of smallest order in $\{T \in \mathcal{T}_G : \ bw(T) \leq bw(T') \ \forall T' \in \mathcal{T}_G\}$.

Note that this definition is consistent with Definition 2.1.

**Theorem 3.12** *For a tree $G$ and a hereditary class of trees $\mathcal{T}$, the $\mathcal{T}$-centroid of $G$ is unique unless both $\mathcal{T} = \{K_1\}$ and $|M(G)| = 2$.*

**Corollary 3.13** *Let $G$ be a tree, and let $D$ be a positive integer. The $T_D$-centroid of $G$ is unique and contains the $T_{D'}$-centroids of $G$ for all $1 \leq D' \leq D$.*

The algorithms for finding $\mathcal{T}$-centroids are not quite the same as for finding $\mathcal{T}$-centers.

**Theorem 3.14** *Let $G$ be a tree and let $G[Y_k]$ be its unique minimal subtree with branchweight at most $k$, for $0 \leq k \leq bw(M(G))$. There is a linear time algorithm that finds all $Y_k$ for $0 \leq k \leq bw(M(G))$.*

**Proof.** We show that we can compute all $Y_k$ in linear time in a similar way as done for the sets $X_k$. First, $M(G)$ can be found in linear time [5]. If $M(G) = \{x\}$, we let $x$ be the root of the tree $G$. If $M(G) = \{x, y\}$, we contract $y$ to $x$ and then let $x$ be the root of the adjusted tree (and we still refer to the adjusted tree as $G$). Note that now $bw(M(G)) = bw(x)$. In the resulting rooted tree, each non-root vertex $v$ is itself the root of a unique maximal tree $G_v$, induced by all vertices reachable from $x$ *via* $v$. We compute $|V(G_v)|$ for all $v$ recursively via a depth-first search. Note that $|V(G_v)| \leq bw(M(G))$ for each non-root vertex $v$.

For each $1 \leq k \leq bw(M(G))$ we create a list, and place each non-root vertex $v$ in the list with $k = |V(G_v)|$ (in linear time). Now let $Y_{bw(M(G))} = M(G)$, and for each $k < bw(M(G))$, let $Y_k = Y_{k+1} \cup \{v\colon |V(G_v)| = k + 1\}$. This can be done in linear time using the lists we set up. Thus, all $Y_k$ are found in linear time. ■

**Corollary 3.15** *Let $G$ be a tree and let $D$ be a positive integer. There is a linear time algorithm for finding the $\mathcal{T}_D$-centroid of $G$. There are similar linear time algorithms for finding the $\mathcal{T}$-centroid if $\mathcal{T}$ is the family of all of any of the following: any finite hereditary class of trees (such as trees of order at most $n$), stars (including $K_1$ and $K_2$), spiders (subdivisions of stars, including all stars), caterpillars (and paths), and lobsters (and caterpillars and paths).*

**Proof.** The $\mathcal{T}_D$-centroid can be found in linear time for the same reasons that the $\mathcal{T}_D$-center can be found in linear time, and the $D = 2$ case resembles the algorithm for finding the path centroid given in [21]. Likewise, we can find the $\mathcal{T}$-centroid in linear time if $\mathcal{T}$ represents any finite hereditary class of tree (such as the trees of order at most $n$), the stars, or the spiders.

Let $\mathcal{C}$ be the hereditary class of all caterpillars and paths. The caterpillar centroid (i.e., $\mathcal{C}$-centroid) is a bit harder to find than the caterpillar center, since a vertex of $Y_k - Y_{k+1}$ may remain a leaf in $Y_{k-1}$ (indeed, $Y_k = Y_{k-1}$ is a possibility). Still, we can begin with the path centroid $P = G[Y_k]$ for some $k$. Gradually we decrease $k$, adding vertices of $Y_{k-1} - Y_k$. Label the non-leaves of $P$ in order: $v_m, \ldots, v_{m+p}$, where $m$ is any integer. If a vertex of $Y_{k-1} - Y_k$ is adjacent to a leaf $u$ of $G[Y_k]$ that is adjacent to some $v_i$ with $m < i < m + p$, then we stop: $G[Y_k]$ is the caterpillar centroid. If the next vertex of $Y_{k-1} - Y_k$ is adjacent to a leaf $u$ of $G[Y_k]$ that is adjacent to $v_m$ or $v_{m+p}$, we relabel $u$ as $v_{m-1}$ or $v_{m+p+1}$ as appropriate, descrease $m$ or increase $p$ by 1, and continue. Thus, we find the caterpillar centroid in linear time. If we continue on from this point we can find the $\mathcal{T}$-centroid where $\mathcal{T}$ is the set of lobsters, in linear time. The process is quite similar to the caterpillar case, except that we act when a vertex is added that is at distance 3 from some $v_i$. Again, the action depends on whether $i$ is an extreme value. ■

Corollary 3.15 did not mention the case where $\mathcal{T}$ is the family of trees with diameter at most $D$, since is not as obvious for $\mathcal{T}$-centroid as it is for $\mathcal{T}$-center. Certainly the formula for $k$ in Corollary 3.9 does not extend to this situation.

## 4 $\mathcal{T}_D$-medians

Now we turn to the $\mathcal{T}_D$-medians. Recall that Slater's path median need not be unique. This is also true for the $\mathcal{T}_D$-medians for any $D \geq 2$, as the following examples illustrate.

First, recall that any median vertex is a $\mathcal{T}_0$-median, and, if there are two median vertices, then the $\mathcal{T}_1$-median is the subgraph on them. If there is only one median vertex $x$ and $|V| > 1$, then the $\mathcal{T}_1$-medians of $G$ are formed by taking $x$ and one neighbor in a largest component of $G - x$. For example, a $k$-star (with $k$ leaves) has $k$ distinct $\mathcal{T}_1$-medians. A $\mathcal{T}_2$-median is just a path median and examples showing that the path median is not unique and need not contain the median are given in [16] and [21].

Next we investigate the $\mathcal{T}_2$-medians of a certain class of trees, in part to motivate the main proof of this section. Let $k \geq 3$, and, for each $1 \leq i \leq k$, let $T_i$ be a either a 4-vertex path or a 5-star, with one leaf labeled $x$. Let $G$ the tree formed by identifying all the vertices labeled $x$. It is not hard to check that a $\mathcal{T}_2$-median must contain $x$. Since a

$\mathcal{T}_2$-median must be a maximal path, it must intersect exactly two components of $G - x$. Let $T$ be such a maximal path. If $T$ does not enter a $T_i$ which is a path, then the vertices of $T_i$ contribute 6 to $s(T)$. Otherwise, $T$ contains $T_i$ and there is no contribution from $T_i$ to $s(T)$. If $T$ does not enter a $T_i$ which is a star, then the vertices of $T_i$ contribute 9 to $s(T)$; otherwise, the vertices of $T_i$ still contribute 3 to $s(T)$. Therefore, no matter which two components of $G - x$ intersect $T$, the status of $T$ is the same, and $T$ is a $\mathcal{T}_2$-median. Thus $G$ may have many $\mathcal{T}_2$-medians. Moreover, if $k \geq 3$ and $G$ has both types of $T_i$, then there are non-isomorphic $\mathcal{T}_2$-medians (paths of different lengths)

For $D \geq 3$, we also have trees with multiple $\mathcal{T}_D$-medians of different isomorphism classes. Let $k \geq D + 1$, and, for $1 \leq i \leq k$, let $T_i$ be either a $(D + 3)$-star with a leaf labeled $x$, or a $D$-star with one leaf labeled $x$, and such that an edge not adjacent to $x$ is subdivided producing exactly one vertex of degree 2. Now let $G$ be the tree formed by identifying all the vertices labeled $x$. (See Fig. 1 for an example.) Any subtree of $G$ not containing $x$ is not a $\mathcal{T}_D$-median, because it can easily be altered to produce a new subtree with smaller status and with maximum degree still at most $D$. A $\mathcal{T}_D$-median cannot intersect every component of $G - x$. A component that does not intersect a $\mathcal{T}_D$-median contributes $2D + 5$ or $2D + 2$ to its status, and the other components contribute 3 or 0 to the status. The "relative cost" of selecting/not selecting each $T_i$ is $2D + 2$. Thus $G$ has at least $\binom{k}{D}$ distinct $\mathcal{T}_D$-medians, and one can easily check that there are $\mathcal{T}_D$-medians of different isomorphism classes as long as $G$ includes both types of $T_i$. In the case that $k = 2D$, one can show that the number of $\mathcal{T}_D$-medians in this example is asymptotically $2^{2D}/\sqrt{\pi D}$ and $n \sim 2D^2$, so the number of $\mathcal{T}_D$-medians is superpolynomial in $n$.

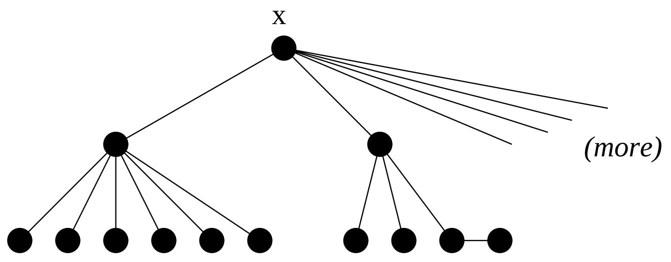

Figure 1: An example with $D = 4$ and $k = 6$, focused on $T_1$ and $T_2$.

Despite all this, the following theorem produces all $\mathcal{T}_D$-medians quickly. The ideas are similar to those in [16]. For any vertex $v$ of $G$, the set of neighbors of $v$ is denoted by $N(v)$. In the proof we need the following definition.

**Definition 4.1** Let $G$ be a tree, let $D$ be a positive integer, let $x$ be a vertex of $G$, and let $\mathcal{T}_D$ be the set of subtrees of $G$ with maximum degree at most $D$. A subtree $T$ of $G$ is a $(D, x)$-*core* in $G$ if $T$ minimizes status among all subtrees in $\mathcal{T}_D$ containing $x$.

**Theorem 4.2** *Let $G$ be a tree, and let $D$ a positive integer. There exists a recursive algorithm that finds all $\mathcal{T}_D$-medians in time that is linear in the order of $G$ plus the number of $\mathcal{T}_D$-medians. There is an algorithm that finds a single $\mathcal{T}_D$-median in linear time.*

**Proof.** Let $x_1, \dots, x_n$ be an ordering of the vertices of $G$ such that $x_i$ is a leaf in $G[\{x_i, \dots, x_n\}]$ for each $1 \leq i < n$. For each $i < n$, let $x_{p(i)}$ be the unique neighbor of $x_i$ with $p(i) > i$. Then for every edge $x_i x_j$, $j = p(i)$ if and only if $j > i$. For each $i < n$, let $V_i$ be the vertex set of the component of $G - x_i x_{p(i)}$ that contains $x_i$. Let $V_n = V$.

The general idea of the algorithm is as follows. Suppose that $T$ is a $\mathcal{T}_D$-median of $G$. Let $i$ be maximum such that $x_i \in V(T)$, then $T \subseteq G[V_i]$. Note that for any tree $T'$ in $G[V_i]$ that contains $x_i$, the status of $T'$ in $G$ is the sum of the status of $T'$ in $G[V_i]$ and the status of $x_i$ in $G - (V_i - x_i)$. Therefore $T$ must have the least status in $G[V_i]$ among all trees in $G[V_i]$ that contain $x_i$ and have maximum degree at most $D$, that is, $T$ is a $(D, x_i)$-core in $G[V_i]$. This directly motivates the first $n$ steps of the algorithm, in which we find all $(D, x_i)$-cores in $G[V_i]$, for $1 \leq i \leq n$. Later in the algorithm we compute the status of these subtrees with respect to $G$. The ones that have minimum status with respect to $G$ are the $\mathcal{T}_D$-medians of $G$.

Suppose that $T$ is a $(D, x_i)$-core in $G[V_i]$. By a similar argument as above, $T$ is the union of $(D, x_i)$-cores in $G[V_k \cup \{x_i\}]$, for some $x_k$ with $p(k) = i$. (See Figure 2.) Note that for each $x_k$ with $p(k) = i$, removing $x_i$ from the $(D, x_i)$-cores in $G[V_k \cup \{x_i\}]$ yields precisely the subtrees of $G[V_k]$ with minimum status in $G[V_k]$ among those that contain $x_k$, have degree at most $D - 1$ at $x_k$, and have degree at most $D$ at every other vertex. We call such a subtree a *$(D, x_k)'$-core*. The difference between this and a $(D, x_k)$-core in $G[V_k]$ is that the $(D, x_k)'$-core minimizes status over subtrees with the additional requirement that the degree at $x_k$ is at most $D - 1$. Thus, in order to use recursion, we keep track of the $(D, x_k)'$-cores as well for all $k < n$. In order to decide which of them might be in $T$, we compute the relative cost of selecting or not selecting each $(D, x_k)'$-core to be part of a $(D, x_i)$-core in $G[V_i]$, as in the two examples given prior to this theorem. In particular, for each $x_k$ with $p(k) = i$, note that if $x_k \notin V(T)$, then an amount equal to the status of $x_i$ in $G[V_k \cup \{x_i\}]$ is contributed to the status of $T$ in $G[V_i]$. Hence, for each $i < n$, we wish to know the status of $x_{p(i)}$ in $G[V_i \cup \{x_{p(i)}\}]$. We denote this by $s(i)$. Also, for $1 \leq i \leq n$:

let $s_D(i)$ be the status of a $(D, x_i)$-core in $G[V_i]$,

let $\mathcal{S}^*_D(i) = \{x_k \colon p(k) = i$ and $x_k$ is in every $(D, x_i)$-core in $G[V_i]\}$, and

let $\mathcal{S}_D(i) = \{x_k \colon p(k) = i$ and $x_k$ is in a $(D, x_i)$-core in $G[V_i]\} - \mathcal{S}^*_D(i)$.

The $(D, x_i)'$-cores will also be computed recursively, so for $1 \leq i < n$:

let $s'_D(i)$ be the status of a $(D, x_i)'$-core in $G[V_i]$,

let ${\mathcal{S}^*_D}'(i) = \{x_k \colon p(k) = i$ and $x_k$ is in every $(D, x_i)'$-core$\}$, and

let $\mathcal{S}'_D(i) = \{x_k \colon p(k) = i$ and $x_k$ is in a $(D, x_i)'$-core$\} - \mathcal{S}^*_D(i)$.

We keep track of the $(D, x_i)$-cores in $G[V_i]$ and the $(D, x_i)'$-cores indirectly via these parameters, because there might be many more of these for various $i$ than there are $\mathcal{T}_D$-medians in $G$. Generating them all explicitly could conflict with the desired running time.

The algorithm begins with recursive steps $1 \leq i < n$, in step $i$ computing $|V_i|$, $s(i)$, $s'_D(i)$, ${\mathcal{S}^*_D}'(i)$, $\mathcal{S}'_D(i)$, $s_D(i)$, $\mathcal{S}^*_D(i)$, and $\mathcal{S}_D(i)$. In step $n$ we compute $s_D(n)$, $\mathcal{S}^*_D(n)$, and

$\mathcal{S}_D(n)$. Note that since $G[V_n] = G$, this computes the status of a $(D, x_n)$-core in $G$. Afterwards, we compute the status of all $(D, x_i)$-cores in $G$ as $i$ decreases from $n-1$ to 1. Finally, we show how all this is used to obtain the $\mathcal{T}_D$-medians of $G$.

In Step $i$ for $1 \leq i < n$, let $N_i$ denote $N(x_i) - x_{p(i)}$ $(= N(x_i) \cap V_i = \{x_k \colon p(k) = i\})$. Note that $N_i$ is empty whenever $x_i$ is a leaf in $G$. For the non-leaves $x_i$, step $i$ will run in time proportional to $|N_i|$ for $i < n$, and the $n^{th}$ step will take time proportional to $|N(x_n)|$, so the first $n$ steps take $O(n)$ time overall.

In the case that $x_i$ is a leaf in $G$, then $|V_i| = 1$, $s(i) = 1$, $s'_D(i) = s_D(i) = 0$, $\mathcal{S}^*_D{}'(i) = \mathcal{S}'_D(i) = \mathcal{S}^*_D(i) = \mathcal{S}_D(i) = \emptyset$. Now suppose that $x_i$ is not a leaf in $G$. Clearly $|V_i| = 1 + \sum_{x_k \in N_i} |V_k|$ and $s(i) = |V_i| + \sum_{x_k \in N_i} s(k)$.

Next we find $s_D(i)$, $\mathcal{S}^*_D(i)$, and $\mathcal{S}_D(i)$. If $|N_i| \leq D$, then $\mathcal{S}^*_D(i) = N_i$, $\mathcal{S}_D(i) = \emptyset$, and $s_D(i) = \sum_{x_k \in N_i} s'_D(k)$. Otherwise, $|N_i| > D$. Then $\mathcal{S}^*_D(i)$ cannot include all of $N_i$. For each $x_k \in N_i$, let $m_k = s(k) - s'_D(k)$ which represents the "relative cost" of not having $x_k$ in $\mathcal{S}^*_D(i)$. Now we place vertices of $N_i$ into $\mathcal{S}^*_D(i)$ and $\mathcal{S}_D(i)$ as follows:

Repeatedly consider all $x_k \in N_i$ with largest $m_k$. If adding these to $\mathcal{S}^*_D(i)$ will not make $|\mathcal{S}^*_D(i)|$ larger than $D$, then do so, and remove that value $m_k$ from consideration (and repeat). If adding all $x_k \in N_i$ with currently largest $m_k$ to $\mathcal{S}^*_D(i)$ would make $|\mathcal{S}^*_D(i)|$ larger than $D$, then let $\mathcal{S}_D(i)$ be the set of $x_k \in N_i$ with currently largest $m_k$, and stop. Note that for any set $N'$ of size $D$ which contains $\mathcal{S}^*_D(i)$ and is contained in $\mathcal{S}^*_D(i) \cup \mathcal{S}_D(i)$, we have that $\sum_{x_k \in N'} s'_D(k) + \sum_{x_k \in N_i - N'} s(k)$ is constant; let $s_D(i)$ be this value. Also note that for any other $N' \subseteq N_i$ of size at most $D$, the sum $\sum_{x_k \in N'} s'_D(k) + \sum_{x_k \in N_i - N'} s(k)$ is larger than $s_D(i)$. Thus, $s_D(i)$, $\mathcal{S}^*_D(i)$, and $\mathcal{S}_D(i)$ have been computed correctly. Note that it takes $O(N_i)$ time to select the vertices $x_k \in N_i$ with currently largest $m_k$, and this is done at most $D$ times. $D$ is constant so the whole step takes $O(N_i)$ time.

Finding $s'_D(i)$, $\mathcal{S}^*_D{}'(i)$, and $\mathcal{S}'_D(i)$ is very similar. The difference is that instead of ceasing to add $x_k \in N_i$ with largest $m_k$ to $\mathcal{S}^*_D(i)$ when it would make $|\mathcal{S}^*_D(i)|$ larger than $D$, we stop adding $x_k \in N_i$ with largest $m_k$ to $\mathcal{S}^*_D{}'(i)$ when it would make $|\mathcal{S}^*_D{}'(i)|$ larger than $D-1$ (or equal to $|N_i|$). Then let $\mathcal{S}'_D(i)$ be the set of $x_k \in N_i$ with currently largest $m_k$, and let $s'_D(i) = \sum_{x_k \in N'} s'_D(k) + \sum_{x_k \in N_i - N'} s(k)$ where $N'$ is any set of size $D-1$ which contains $\mathcal{S}^*_D{}'(i)$ and is contained in $\mathcal{S}^*_D{}'(i) \cup \mathcal{S}'_D(i)$. Obviously, any reasonable implementation would combine these three computations with the previous three with the same $i$.

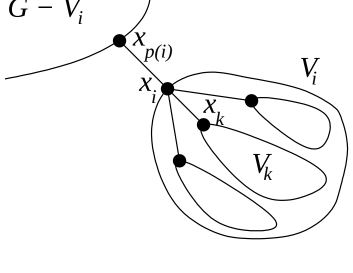

Figure 2: Something to look at for Steps 1 to $n$

In Step $n$ we compute $s_D(n)$, $\mathcal{S}^*_D(n)$, and $\mathcal{S}_D(n)$. These are computed exactly as $s_D(i)$, $\mathcal{S}^*_D(i)$, and $\mathcal{S}_D(i)$ are computed for $i < n$ if we define $N_n$ to be $N(x_n)$.

It still remains to compute the status *with respect to* $G$ of each $(D, x_i)$-core of $G[V_i]$. Toward this purpose, let $s'(i)$ denote the status of $x_i$ in $G - (V_i - x_i)$. We compute $s'(i)$ as $i$ decreases from $n-1$ to 1. Observe that $s'(i)$ is the sum of $s'(p(i))$, all $s(j)$ such that $p(j) = p(i)$ and $j \neq i$, and $|V - V_i|$. (See Fig. 2 for an example of $s'(k)$.) Since $|V - V_i| = n - |V_i|$ and $s'(p(i))$ is computed before $s'(i)$, we are able to compute $s'(i)$ for all $1 \leq i < n$.

For $1 \leq i < n$, let $s^*(i) = s_D(i) + s'(i)$ and let $s^*(n) = s_D(n)$. Then for all $1 \leq i < n$, the value $s^*(i)$ is the status *measured in* $G$ of a $(D, x_i)$-core in $G[V_i]$. Let $s(G) = \min_{1 \leq i \leq n} s^*(i)$. Then, by our initial argument, $s(G)$ is the status of a $\mathcal{T}_D$-median of $G$.

Finally, we will show how to obtain every $\mathcal{T}_D$-median of $G$. Begin with any $x_i \in V$ for which $s^*(i) = s(G)$. Add $\min\{D, |N_i|\}$ neighbors of $x_i$ from $\mathcal{S}_D^*(i) \cup \mathcal{S}_D(i)$, including all of $\mathcal{S}_D^*(i)$. Now starting from each $x_k$ just chosen, recursively add $\min\{D-1, |N_i|\}$ vertices from $\mathcal{S}_D^*{}'(k) \cup \mathcal{S}_D'(k)$ including all of $\mathcal{S}_D^*{}'(k)$. The choices made determine which of the $\mathcal{T}_D$-medians of $G$ is obtained, and this $\mathcal{T}_D$-median is obtained in linear time. By branching the procedure to follow through with all possible choices, we obtain all $\mathcal{T}_D$-medians. Note that for a given $\mathcal{T}_D$-median, if $x_i$ is its vertex of maximum index, then this $\mathcal{T}_D$-median is a $(D, x_i)$-core but not a $(D, x_j)$-core for any $j \neq i$. Hence this $\mathcal{T}_D$-median will be only be produced once by this process, justifying the claimed running time for producing all $\mathcal{T}_D$-medians. Finally, if we add all of $\mathcal{S}_D^*(i) \cup \mathcal{S}_D(i)$ every time, we obtain the set of all vertices that are are contained in $T_D$-medians of $G$. ■

**Remark 4.3** If we knew in advance that some vertex of $G$ is in every $\mathcal{T}_D$-median, then we could order the vertices so that $x_n$ is that vertex. Then every $\mathcal{T}_D$-median would actually be a $(D, x_n)$-core in $G$, which would simplify things somewhat. The most obvious candidate would be a median vertex, except that Morgan and Slater [16] showed that sometimes the path median of a tree does not contain a median vertex. However, Slater [22] has found a different sort of vertex which is contained in every path median, so perhaps there is hope for $\mathcal{T}_D$-medians as well. On the other hand, one might try to find an example of a tree $G$ that has two $\mathcal{T}_D$-medians that do not intersect one another.

## 5 Conclusions

This paper unifies many types of central substructures of trees under the definitions of $\mathcal{T}$-center and $\mathcal{T}$-centroid, subsuming previous definitions and algorithms. It also deals with many other potential generalizations, since our work immediately applies whenever $\mathcal{T}$ is a hereditary class of trees. For many choices of $\mathcal{T}$, one can follow our model and show how to find the $\mathcal{T}$-center and $\mathcal{T}$-centroid in linear time. This leads to the question: are there linear time algorithms for finding the $\mathcal{T}$-center and $\mathcal{T}$-centroid for *any* hereditary class of trees $\mathcal{T}$, and, if so, can the algorithms be described in a unified manner? The answer to the first part of the question would be 'Yes' if, for every hereditary class of trees $\mathcal{T}$, there is a sufficiently fast recognition algorithm to test whether a subtree $T$ of an arbitrary tree $G$ is in $\mathcal{T}$. It might help to have a nice alternative characterization of a hereditary class of trees.

Another direction to pursue would be to define a $\mathcal{T}$-median in the obvious way, and to find fast algorithms for finding one or all $\mathcal{T}$-medians in a tree $G$, for any hereditary class of trees $\mathcal{T}$, or merely for special hereditary classes of trees $\mathcal{T}$. Yet another possibility is to see whether our definitions are related to disconnected central substructures, e.g., is it true that a $p$-median [7, 8, 12] and $p$-core [2, 24] must always be contained in a $\mathcal{T}_D$-center for $D$ sufficiently large (as a function of $p$)? If so, one might be able to use a $\mathcal{T}_D$-median (which we can find quickly) to quickly find a $p$-median or $p$-core. (See [24] for a quick survey on the best-known algorithms for finding a $p$-median and $p$-core.) One might try to find a $p$-center quickly by a similar strategy, although there is a (rather complicated) linear time algorithm already [4]. Finally, there are vertex- and edge-weighted versions of these problems, versions of the problems where a portion of an edge can be in a central substructure, and the more general situation of when the host graph is not a tree.

**Acknowledgements**

We thank Buck McMorris for organizing the 2003 DIMACS Reconnect Workshop at Illinois Institute of Technology that initiated this study, and for his suggestion of investigating caterpillar centers, which eventually led us to come up with hereditary classes of trees.

## References

[1] R.I. Becker, Y.I. Chang, I. Lari, A. Scozzari, G. Storchi, *Finding the l-core of a tree*, Third ALIO-EURO Meeting on Applied Combinatorial Optimization (Erice, 1999). Discrete Appl. Math. 118 (2002), no. 1-2, 25–42.

[2] R. I. Becker and Y. Perl, *Finding the two-core of a tree*, Discrete Appl. Math. **11**(2) (1985), pp. 103–113.

[3] E. J. Cockayne, S. T. Hedetniemi, and S. L. Hedetniemi, *Linear Algorithms for Finding the Jordan Center and Path Center of a Tree*, Trans. Sci. **15** (1981), pp. 98–114.

[4] G.N. Frederickson, *Parametric Search and Locating Supply Centers in Trees*, (Algorithms and Data Structures, 2nd Workshop, WADS '91, Ottawa, Canada, August 1991), LNCS 519, Springer-Verlag, pp. 299-319.

[5] A. J. Goldman, *Optimal center location in simple networks*, Trans. Sci. **5** (1971), pp. 212–221.

[6] A. J. Goldman, *Minimax location of a facility on a network.* Trans. Sci. **6** (1972), pp. 407–418.

[7] S. L. Hakimi, *Optimum locations of switching centers and absolute centers and medians of a graph*, Operations Res. **12** (1964), pp.450-459.

[8] S. L. Hakimi, *Optimum locations of switching centers and absolute centers in a communication network and some related graph-theoretic problems*, Operations Res. **13** (1965), pp.462–475.

[9] G. Y. Handler, *Minimax Location of a Facility in an Undirected Tree Graph*, Trans. Sci. **7** (1973), pp. 287–293.

[10] K. S. Holbert, *A note on graphs with distant center and median*, Recent Studies in Graph Theory, V.R.Kulli (editor), Vishna, Gulbarza, India, 1989, pp.155–158.

[11] C. Jordan, *Sur les assemblages de lignes*, J. Reine Agnew. Math. **70**, 185–190 (1869).

[12] S. L. Hakimi and O. Kariv, *An algorithmic approach to network location problems. II: The p-medians*, Siam. J. Appl. Math. **37**(3) (1979), pp. 539–560.

[13] F. R. McMorris, Personal Communication, 2003.

[14] F. R. McMorris and K. B. Reid, *Central k-trees in Trees*, Congress. Numer. **124** (1997), pp. 139–143.

[15] E. Minnieka, *The optimal location of a path or tree in a tree network*, Networks **15** (1985), 309–321.

[16] C. A. Morgan and P. J. Slater, *A Linear Algorithm for the Core of a Tree*, J. Algorithms **1** (1980), 247–258.

[17] J. Nieminen, *Centrality, convexity and intersections in graphs*, Bull. Math. Soc. Sci. Math. R. S. Roumanie (N.S.) 28(76) (1984), no. 4, 337–344.

[18] M.J. Pelsmajer and J. Pierce, *A linear time algorithm for finding a vertex p-center in an unweighted tree*, in process.

[19] S. Peng, A.B. Stephens, Y. Yesha, *Algorithms for a core and k-tree core of a tree*, J. Algorithms 15 (1993), no. 1, 143–159.

[20] A. Shioura and T. Uno, *A linear time algorithm for finding a k-tree core*, J. Algorithms 23 (1997), no. 2, 281–290.

[21] P. J. Slater, *Locating central paths in a graph*, Transportation Sci., **16** (1982), 1–18.

[22] P. J. Slater, *Centrality of paths and vertices in a graph: cores and pits*, The theory and applications of graphs (Kalamazoo, Mich., 1980), pp. 529–542, Wiley, New York, 1981.

[23] Srivastava, Saurabh and Ghosh, R. K., *Distributed algorithms for finding and maintaining a k-tree core in a dynamic network.*, Inform. Process. Lett. 88 (2003), no. 4, 187–194.

[24] B.-F. Wang, *Finding a 2-core of a tree in linear time.*, SIAM J. Discrete Math. **15**(2) (2002), pp.193–210.

[25] B. Zelinka, *Medians and Peripherians of Trees*, Arch. Math. (Brno), 87–95 (1968).