Upper and Lower Bounds for Optimal Tree Partitions†

Leonidas Palios
The Geometry Center
University of Minnesota
Minneapolis, MN 55454

Abstract: The work in this paper is motivated by a common graph partitioning problem, which asks for the partition of the node set of a given graph G into two sets of prespecified cardinalities that minimizes the *cutsize* (i.e., the number of G's edges with one endpoint in either set). The problem finds applications in VLSI and networking. We consider in particular the case where G is an arbitrary tree. We prove that for an n-node tree whose nodes have no more than $d{+}1$ neighbors ($d \geq 2$), there exists a partition of the tree nodes into two sets V_1 and V_2 of desired sizes, whose cutsize is no more than $(d-1)\ \log_d(\min\{|V_1|, |V_2|\}) + 1$; in case of a *bisection*, that is, if $|V_1| = \lfloor n/2 \rfloor$, this bound becomes $(d-1)\ \log_d n + 1$. Additionally, for $d \geq 3$, we exhibit families of trees, whose bisection involves cutsizes no less than $\lfloor (d-1)/2 \rfloor\ (\log_d n - \log_d \log_d n) + \Theta(d)$, nearly a factor of 2 away from the corresponding upper bound. Thus, the cutsize of the bisection of such a tree is $\Theta(d\ \log_d n)$, which is $\Theta(\log_2 n)$ if d is a constant, and $\Theta(n)$ if $d = \Theta(n)$. Finally, for $d = 2$, a completely different approach yields a slightly worse lower bound of $\lceil \log_2(n+1) \,/\, (2\ \log_2 \log_2(n+1)) \rceil$.

† This work has been supported by grants from the National Science Foundation (NSF/DMS-8920161), the Department of Energy (DOE/DE-FG02-92ER25137), Minnesota Technology, Inc., and the University of Minnesota.

1. Introduction.

The work in this paper is motivated by a common *graph partitioning* problem, which can be stated as follows: given a graph G of n nodes and a positive integer $m < n$, find a partition of G's node set into two sets of sizes m and $n - m$ that minimizes the number of G's edges with one endpoint in either set. The problem is very interesting, as it both incorporates the minimization of a resource and a load balancing mechanism through appropriate values of m, and finds applications in VLSI (placement of components) and networking (minimization of communication cost). The most popular among all partitions of a graph is its *bisection*, where the cardinalities of the two partition sets differ by no more than 1. Due to the need for good graph bisections, especially in graph algorithms based on the divide-and-conquer paradigm, fast algorithms that yield optimal or near-optimal bisections are very important. Unfortunately, the graph bisection problem on general graphs is proven to be NP-complete [4]; this motivated research towards finding heuristics with good average case behavior ([1], [2], [5], [6]). It is yet unknown whether the graph bisection problem remains NP-complete when restricted to the class of planar graphs, whereas optimal bisections of trees can be computed in $O(n^3)$ time using dynamic programming. Returning to the problem of partitioning a graph into two sets of prespecified cardinalities, few results are known. The most recent one is an algorithm due to Bui and Peck [3] to compute an optimal partition of a planar graph of n vertices into sets of sizes m and $n - m$, where $m \in [0, n]$. The algorithm runs in $O(b^2 n^3 2^{4.5b})$ time, where b is the number of edges with one endpoint in either set of the computed optimal partition; since b can be as large as $\sqrt{n}$, the algorithm is not necessarily polynomial.

In this paper, we are interested in the graph partitioning problem applied to the case of trees. Although rather restricted, the problem is still important as many graphs in practice often reduce to trees. For instance, graphs that illustrate the interdependence of the procedures of a program (in terms of procedure calls) frequently exhibit a tree structure; then, the problem of assigning procedures to two remote sites (each being able to handle a certain number of them without being unacceptably slow) while at the same time minimizing the communication cost reduces to optimally partitioning a tree. Our work focuses on establishing upper and lower bounds for the minimum cutsize of a partition of the nodes of a tree into two sets of prespecified cardinalities, the cutsize of a partition being the number of tree edges with one endpoint in either partition set. First, we prove an upper bound in the minimum cutsize of a partition: we describe a way to collect sets of the trees nodes that eventually form the desired partition. The method implies that, for an n-node rooted tree whose nodes have no more than d children, there exists a partition of the tree nodes into two sets V_1 and V_2 of desired sizes, whose cutsize is no more than $(d-1)\ \log_d \min\{|V_1|, |V_2|\}+1$. Note that for a bisection this bound becomes $(d-1)\ \log_d n + 1$. We also establish that, for $d \geq 3$, the optimal bisection of appropriate families of trees exhibits cutsizes no less than $\lfloor (d-1)/2 \rfloor\ (\log_d n - \log_d \log_d n) + \Theta(d)$; in light of the earlier upper bound, the cutsize of the bisection of such trees is $\Theta(d\ \log_d n)$, which is $\Theta(\log_2 n)$ if d is a constant, and $\Theta(n)$ if $d = \Theta(n)$. In case $d = 2$, we use a completely different approach to prove that there exists an optimal partition of an n-node tree that exhibits cutsizes at least $\lceil \log_2(n+1) / (2 \ \log_2 \log_2(n+1)) \rceil$.

The paper is structured as follows. In Section 2, we present the necessary terminology that we use throughout the paper. In Section 3, we establish the upper bound on the minimum cutsize of a tree partition in terms of the sizes of the partition sets and parameters of the tree. A lower bound is proved in Section 4 by showing that bisections of specific trees admit large cutsizes. Section 5 concludes the paper with the final remarks and some open questions.

2. Terminology.

The notion of a tree is one of the most fundamental notions in computer science; its definition and properties can be found in nearly every introductory book on graph theory. Below, we quickly review the basic terminology. A *tree* is a collection of *nodes* connected by *edges* such that for every pair of nodes u and v, there exists a unique *path* connecting them, that is, a sequence $(u = w_1, w_2, \ldots, w_k = v)$ of nodes such that there is an edge between every pair w_i and w_{i+1} $(1 \le i < k)$, and $w_i \neq w_j$ for $i \neq j$. In case a specific node of the tree has been selected as the *root*, the tree is called *rooted*; otherwise it is called *unrooted*. Since any unrooted tree can become rooted if one of its nodes is designated as its root, we will restrict our attention to rooted trees.

A rooted tree can be displayed in a layered fashion along the vertical direction as follows (Figure 1). The highest or *0-th level* is occupied by the root only. The *first level* contains all nodes of the tree that are connected to the root by an edge. In general, the *i-th level* contains all nodes that are connected to a node in the $(i-1)$-st level, and do not belong to the $(i-2)$-nd level. (The latter requirement ensures that each node will be displayed once, and that the layering process will eventually terminate.) If a node belongs to the i-th level, we say that its *depth* is i. The maximum among the depths of all the tree node defines the *height* of the tree (for instance, the height of the tree in Figure 1 is 4). Additionally, the nodes u_i of the j-th level that are connected to a node v in the $(j-1)$-st level by means of an edge are called v's *children*; v is called the *parent* of the u_is. If a node has no children, it is called a *leaf* of the tree. A non-leaf node of the tree is called *internal*. The tree in Figure 1 has 6 leaves and 6 internal nodes. The *subtree* rooted at a node v is the portion of the tree that contains v when the edge between v and its parent is removed.

In case no node of the tree has more than d children, we say that the tree is a *d-tree*. For minimality reasons, we assume that there exists a node in a d-tree that has exactly d children. Finally, we say that a d-tree is *complete* if each of its internal nodes has exactly d children, and all the leaves belong to the same (lowest) level (Figure 2 shows a complete 2-tree of height 3). It is an easy exercise to prove that the i-th level of a complete d-tree contains exactly d^i nodes; if its height is h, the entire tree contains a total of $(d^{h+1}-1)/(d-1)$ nodes.

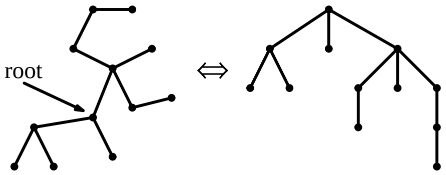

Figure 1

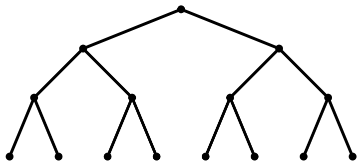

Figure 2

We now introduce the terminology and our notation regarding partitions. For $1 \leq m \leq n-1$, an *$(m, n-m)$-partition* of a graph G (or a tree, in particular) of n nodes is a partition of the node set of G into two sets with cardinalities m and $n-m$; therefore, there exist exactly $\lfloor n/2 \rfloor$ *different* partitions, namely, partitions into sets of 1 and $n-1$, 2 and $n-2$, ..., $\lfloor n/2 \rfloor$ and $\lceil n/2 \rceil$ nodes. A given partition of the node set of G uniquely defines the set G's edges with one endpoint in either partition set, which is called the *cutset* of the partition; the size of the cutset is the *cutsize* of the partition. Conversely, each subset E of tree edges defines a unique partition of the nodes of the tree (note that the removal of any edge disconnects the tree), the underlying assumption being that the nodes incident upon each edge in E do not belong to the same partition set; the cutsize of this partition equals the cardinality of the set E. The above observations exhibit a special correspondence between partitions and sets of tree edges. For a specific m, there are $\binom{n}{m}$ ways to form the partition sets of an $(m, n-m)$-partition, and the corresponding cutsizes may vary substantially; the partition sets that exhibit the minimum cutsize define the *optimal* $(m, n-m)$-partition.

We close this section by noting that in the following we ignore the trivial case of a tree of 1 node, as well as 1-trees (chains), where all possible node set partitions can be achieved at cutsize 1. We therefore deal with d-trees ($d \geq 2$) of $n \geq 2$ nodes.

3. The Upper Bound.

In this section, we establish an upper bound on the cutsize of the optimal $(m, n-m)$-partition of an n-node d-tree for all $m = 1, \ldots, n-1$. It should be obvious that if tree nodes have $\Theta(n)$ children, an $(m, n-m)$-partition may necessitate cutsets whose sizes are as large as $\Theta(m)$. Simply consider a tree that consists of a root node with $n-1$ children. Clearly, an $(m, n-m)$-partition of such a tree requires cutsets of at least $\min\{m, n-m\}$ edges. However, for trees whose nodes have at most a constant number of children, the situation is different. Indeed, we prove below that for a d-tree of n nodes, for constant d, there is always an $(m, n-m)$-partition of the set of tree nodes $\big($for all $m \in \{1, 2, \ldots, n-1\}\big)$ whose cutsize is $O(\log_2 m)$.

MacGregor [7] proves an upper bound of $\tau(d, n)$ on the cutsize of an optimal bisection of a d-tree of n nodes, where

$$\tau(d, n) = \begin{cases} \frac{1}{2} + 2\log_3 n, & \text{if } d = 2, 3; \\ \frac{1}{2} + \frac{d+2}{2}\log_{d+1} n, & \text{if } d \geq 4. \end{cases}$$

Unfortunately, his method does not seem to extend to the case of $(m, n-m)$-partitions for arbitrary m, and, even if it does, it is likely to give rise to a $\log_d n$ term in the bound. This is not desirable, since m may very well be a constant in which case the estimated bound is extremely pessimistic.

We first mention the following two important lemmas.

Lemma 3.1. *Given j positive integers m_i ($j \geq 2$) such that $m_i < m$ for all $i = 1, \ldots, j$ and $\sum_{i=1}^{j} m_i > m$, there exist k of these integers, say, $m_{i_1}, \ldots, m_{i_k}$ ($1 \leq k < j$), such that $\frac{k}{k+1} m < \sum_{t=1}^{k} m_{i_t} < m$.*

Proof: The proof proceeds inductively on j. For $j = 2$, we have $m_1 + m_2 > m$, which easily implies that at least one of the m_is is larger than $m/2$. The additional premise that $m_i < m$ for $i = 1, 2$ yields that the lemma holds for $k = 1$. Let us assume now that the lemma holds for all integers j such that $2 \leq j \leq j_0$. We will prove that it also holds for $j = j_0 + 1$. So, consider $j = j_0 + 1 \geq 3$ integers m_i such that $m_i < m$ and $\sum_{i=1}^{j} m_i > m$ for some $m > j$. Then, we can find $j - 1$ integers among the m_is whose sum exceeds $(j-1)m/j$. If not, then each of the j combinations of $j - 1$ integers out of the m_is should add up to at most $(j-1)m/j$. By adding all these combinations, we get

$$(j-1) \sum_{i=1}^{j} m_i \ \leq \ j \frac{(j-1)m}{j} \iff \sum_{i=1}^{j} m_i \ \leq \ m,$$

which leads to contradiction. So, there exist $j-1$ integers among the m_is (and let us assume, without loss of generality, that these are the integers $m_1, \ldots, m_{j-1}$) which add up to more than $(j-1)m/j$. If this sum is less than m, then the lemma holds for $k = j - 1$; otherwise, we have $j - 1 = j_0 \geq 2$ integers $m_1, \ldots, m_{j-1}$, each of which is less than m, and whose sum exceeds m. Then, the inductive hypothesis implies that, for some integer k such that $1 \leq k < j - 1$, there exist k integers among $m_1, \ldots, m_{j-1}$ whose sum is larger than $km/(k+1)$ but less than m, as desired. ∎

Lemma 3.2. *Consider the function $s : Z_+ \to Z_+$ (Z_+ is the set of positive integers) defined as follows:*

$$s(h) \ \leq \ h, \qquad \text{if } 1 \leq h \leq d$$

$$s(h) \ = \begin{cases} h', & \text{for some integer } h' : \ 1 \leq h' \leq d; \\ s(h') + k, & \text{for integers } h', k : \ 1 \leq h' < \frac{h}{k+1}, \ 1 \leq k < d; \end{cases} \qquad \text{if } h > d$$

where d is an integer larger than 1. Then, $s(h) \leq (d-1) \ \log_d h + 1$ for all $h \geq 1$.

Proof: We use induction on h. Clearly $1 \leq (d-1) \ \log_d 1 + 1$, and the lemma holds for $h = 1$. For the inductive hypothesis, we assume that $s(h) \leq (d-1) \ \log_d h + 1$ for all integers h such that $1 \leq h \leq h_0$. We prove next that the previous inequality also holds for $h = h_0 + 1 > 1$. We distinguish two cases:

1. If $h = h_0 + 1 \le d$, we need only show that $(d-1)\ \log_d h + 1 \ge h$.
$$(d-1)\ \log_d h + 1 \;=\; \frac{d-1}{\ln d}\ \ln h + 1 \;\ge \frac{h-1}{\ln h}\ \ln h + 1 \;=\; h,$$
since the function $f(x) = (x-1)/\ln x$ is strictly increasing for $x > 1$.

2. Suppose $h = h_0 + 1 > d$. Since $(d-1)\ \log_d h + 1 \ge d$ is trivially true for $h > d$, the treatment of this case will be complete if we show that $s(h') + k \le (d-1)\ \log_d h + 1$ for all integers h' and k such that $1 \le h' < h/(k+1)$, $1 \le k < d$. Since $h/(k+1) < h$ for all $h > 0$ and $k \ge 1$, then $h' < h$, and the inductive hypothesis implies that $s(h') \le (d-1)\ \log_d h' + 1$; this in turn is less than $(d-1)\ \log_d\bigl(h/(k+1)\bigr) + 1$, since $h' < h/(k+1)$. Thus,
$$\begin{aligned} s(h) = s(h') + k &\le (d-1)\ \log_d \frac{h}{k+1} + 1 + k \\ &\le (d-1)\ \log_d h - (d-1)\frac{\ln(k+1)}{\ln d} + k + 1 \\ &\le (d-1)\ \log_d h + 1, \end{aligned}$$
for $d > k \ge 1$; the function $f(x) = (x-1)/\ln x$ is strictly increasing for $x > 1$. ∎

We are now ready to establish the desired upper bound.

Theorem 3.1. *Let T be a d-tree ($d \ge 2$) of size $n \ge 2$. Then, for any positive integer $m < n$, we can always find an $(m, n-m)$-partition of the node set of T whose cutsize is at most*
$$\left\lfloor (d-1)\ \log_d\Bigl(\min\{m, n-m\}\Bigr) \right\rfloor + 1.$$

Proof: Let us assume without loss of generality that $m = \min\{m, n-m\}$. Our attempt will be to collect a number of disjoint subtrees of T whose sizes add up to m, and prove that the number of these subtrees does not exceed $(d-1)\ \log_d m + 1$, which will imply the statement of the lemma. We distinguish two cases.

1. In case $m \le d$, we can find m tree nodes of T such that there exist at most m edges connecting any of them to any of the remaining nodes. We need only find a leaf (there always exists a leaf in a tree of $n \ge 2$ nodes), remove it from the tree, and apply the same procedure again $m-1$ more times. The set of removed nodes satisfies the previous condition, that is, the cutsize of the corresponding $(m, n-m)$-partition is at most m.

2. Consider now the case $m > d$. Let us label each of T's nodes with the size of the subtree rooted at it, and let u be the node whose label is the minimum integer that exceeds m. (Note that we can always find such a tree node; the label of T's root is n, which exceeds m, since $0 < m < n$.) Let $j \ge 1$ denote the number of u's children (since $m \ge 1$, u's label is at least equal to 2, and

thus u has at least one child), whose labels are respectively denoted by t_i ($i = 1, \ldots, j$). Obviously, $1 + \sum_{i=1}^{j} t_i \geq m+1$, and $t_i \leq m$ ($i = 1, \ldots, j$), otherwise this fact would contradict the definition of u. If $t_i = m$ for some i or $\sum_{i=1}^{j} t_i = m$, we end up with either 1 or $j \leq d$ subtrees of T whose sizes add up to exactly m, and these subtrees form one of the partition sets. If $t_i < m$ and $\sum_{i=1}^{j} t_i > m$, then note that $j \geq 2$ (the inequalities $t_1 > m$ and $t_1 < m$ do not hold simultaneously), and Lemma 3.1 applies, yielding that for an integer k ($1 \leq k < j$) there exist k disjoint subtrees rooted at u's children, such that the sum of their sizes is larger than $km/(k+1)$ and less than m. Then, we prune these subtrees from T, and save them to be part of the set of m nodes. What is left of T is still a d-tree, and we can apply to it the above procedure to collect the nodes that are needed to make the set of m nodes; their number is less than $m - km/(k+1) = m/(k+1)$.

Therefore, if we denote by $s(h)$ the number of subtrees that are needed to form a set of h tree nodes, the above procedure implies that the function $s(.)$ satisfies the conditions of Lemma 3.2. The desired bound follows directly, given the fact that $s(h)$ assumes integer values for all integer $h \geq 1$. ∎

In particular, for the case of a bisection: we substitute $n/2$ for m and we have

Corollary 3.1: *For any d-tree ($d \geq 2$) of $n \geq 2$ nodes, we can always find a bisection of the tree's node set whose cutsize does not exceed* $\lfloor (d-1) \ \log_d n - (d-1) \ \log_d 2 \rfloor + 1$.

The upper bound on the cutsize of optimal bisections of d-trees implies an $O(\log_2 n)$ upper bound if d is a constant, and an $O(n)$ upper bound if d is $\Theta(n)$.

4. The Lower Bound.

The key issue in proving lower bounds is finding a sufficiently bad instance of the problem at hand. In our case, this translates into finding an infinite family of trees that exhibit partitions necessitating fairly large cutsizes. Additionally, the structure of the trees needs to be characterized by some pattern, which we will exploit in our arguments. It turns out that the family of complete d-trees satisfies both criteria, and it is the one that we are going to use below. It is worth noting, however, that there exist tree families that realize all $\lfloor n/2 \rfloor$ partitions at small cutsizes. Consider, for instance, the tree shown in Figure 3, where all subtrees marked by T_k are trees of k nodes each (not necessarily the same), for some constant $k \geq 1$. Then, it is not difficult to see that the cutsize of any partition of such a tree is at most equal to $\lceil k/2 \rceil + 1$ (see Appendix).

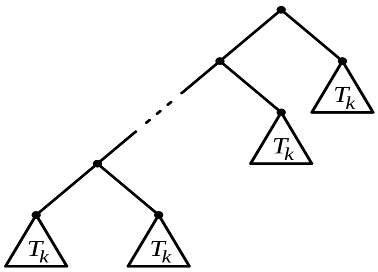

Figure 3

Our approach is the generalization of a method that MacGregor [7] uses in order to prove that the cutsize of an optimal bisection of a complete ternary tree on n nodes is at least equal to $\lceil \log_3 n \rceil - \lceil \log_3 \log_3 n \rceil$. The method extends to complete d-trees, where d is an odd integer. More specifically,

Theorem 4.1. *Let $d \geq 3$ be an odd integer. An optimal bisection of a complete d-tree of $n \geq 2$ nodes has cutsize at least*

$$\frac{d-1}{2}\Big(\log_d n - \log_d \log_d n - 1\Big).$$

Proof: Let $h \geq 1$ denote the height of the tree, so that the total number of tree nodes is $n = (d^{h+1}-1)/(d-1)$. Let us assume that we have already computed an optimal bisection of the set of tree nodes into two sets, say, the set of $\lfloor n/2 \rfloor$ white nodes and the set of $\lceil n/2 \rceil$ black nodes. Then, starting at the top, we *normalize* the bisection by shuffling the subtrees at each level so that all those with black roots are located to the right of all those with white roots; note that this operation does not increase the cutsize of the bisection. It is important to observe that at this point there is no white (black resp.) internal node of the tree whose children are all colored black (white resp.). If, instead, there existed such a node, then we would color it black (white resp.), and we would change the leftmost black (rightmost white resp.) leaf into a white (black) leaf. This color-swapping operation clearly preserves the cardinalities of the two partition sets, and its feasibility stems from the fact that, in any bisection of a complete d-tree, each of the partition sets contains at least one leaf, as the number of internal nodes of such a tree is substantially less than the number of leaves (the ratio of leaves over internal nodes is larger than $(d-1):1$). Moreover, it results in a reduction of at least $d-2 \geq 1$ in the cutsize of the bisection, which contradicts the optimality of the initial bisection.

Now, according to the following lemma (Lemma 4.1), the $(d^i-1)/2$ leftmost nodes of the i-th level ($1 \leq i \leq \lfloor h - \log_d h \rfloor$) are colored white, the $(d^i-1)/2$ rightmost ones are colored black, while the color of the middle one may be either white or black. In other words, at each of the topmost $\lfloor h - \log_d h \rfloor$ levels, the middle tree node has at least $(d-1)/2$ children colored white and at least $(d-1)/2$ children colored black. So, regardless of its color, each such tree node contributes a term at least equal to $(d-1)/2$ in the cutsize of the partition, which therefore is at least $\big(d-1\big)\big(\lfloor h - \log_d h \rfloor\big)/2$. The theorem follows, if we take into account that for $h \geq 1$ and $n = (d^{h+1}-1)/(d-1)$ we have

$$h - log_d h \ > \ \log_d n - \log_d \log_d n - 1.$$

For $h = 1$ ($n = d+1$), the above inequality holds, since $2 > \log_d(d+1)$ for all $d \geq 2$; for $h \geq 2$,

$$h + 1 - log_d h \ > \ (h+1) - log_d(h+1) \ > \ \log_d n - \log_d \log_d n,$$

since the function $f(x) = x - \log_d x$ is strictly increasing for $x \geq 3$, and $h+1 = \log_d((d-1)n+1) > \log_d n$ for $d \geq 2$. ∎

Lemma 4.1. *Let T be a complete d-tree ($d \geq 3$ is an odd integer) of height $h \geq 1$, whose nodes have been colored black and white according to an optimal bisection, and the normalization process described in Theorem 4.1 has been applied. Then, for $1 \leq j \leq h - \log_d h$, the leftmost $(d^j - 1)/2$ tree nodes of the j-th level are white, while the rightmost $(d^j - 1)/2$ ones are black.*

Proof: The proof proceeds inductively on j. The basis is to show that the statement of the lemma is true for $j = 1$, that is, the $(d-1)/2$ leftmost children of the root are colored white, while the $(d-1)/2$ rightmost ones are colored black. We show that the $\big((d-1)/2\big)$-th tree node from the right, let it be t_1, is colored black. Assume that it is colored white, for contradiction. Then, according to the normalization procedure, so are the $(d+1)/2$ children of the root to the left of t_1. Moreover, at least one of t_1's children must be colored white, otherwise we would have a white node with all black children. This implies that t_1's leftmost child, let it be t_2, as well as all the $d(d+1)/2$ nodes on the same level and to the left of t_2 are colored white. The same reasoning can be applied repeatedly at all the lower levels, implying that the i-th level contains at least $d^{i-1}(d+1)/2 + 1$ white nodes, so that the number of white nodes is at least equal to

$$
\begin{aligned}
\sum_{i=1}^{h} \left(d^{i-1} \, \frac{d+1}{2} + 1 \right) &= \frac{d+1}{2} \sum_{i=0}^{h-1} d^i \; + h \\
&= \frac{d+1}{2} \, \frac{d^h - 1}{d-1} + h \\
&= \frac{1}{2} \, \frac{d^{h+1} - 1}{d-1} + \frac{d^h - d}{2(d-1)} + h \\
&= \frac{n}{2} + \frac{d^h - d}{2(d-1)} + h,
\end{aligned}
$$

which is larger than $(n+1)/2$ for all $h \geq 1$; a contradiction. The $\big((d-1)/2\big)$-th tree node of the first level from the right is therefore colored black, and so are the remaining $(d-3)/2$ nodes to its right, according to our normalization process. Hence, the $(d-1)/2$ rightmost tree nodes of that level are colored black. The above argument applies in a similar fashion for the case of the white nodes of the first level, implying that the $(d-1)/2$ leftmost tree nodes of the first level are colored white.

For the inductive step, we assume that for an integer k such that $1 \leq k \leq h - \log_d h - 1)$, the leftmost $(d^i - 1)/2$ tree nodes of the i-th level ($1 \leq i \leq k$) are colored white, while the rightmost $(d^i - 1)/2$ ones are colored black. Let us now consider the $(k+1)$-st level, and let us prove that the rightmost $(d^{k+1} - 1)/2$ tree nodes are colored black; then, the argument can be extended to prove that the leftmost $(d^{k+1} - 1)/2$ tree nodes are colored white. So, suppose for contradiction that the $\big((d^{k+1} - 1)/2\big)$-th tree node of the $(k+1)$-st level from the right is colored white. This implies that the $(d^{k+1} + 1)/2$ tree nodes to its left are colored white as well. Moreover, its leftmost child t_{k+2} is colored white too, otherwise there would exist a white tree node with all black children; then, the $d(d^{k+1} + 1)/2$ nodes to the left of t_{k+2} are also white. The same argument applies to the leftmost child of t_{k+2} and so on so forth all the way down the tree. At the i-th level ($k+1 \le i \le h$), we have at least $d^{i-k-1}\frac{d^{k+1}+1}{2}+1$ nodes that are colored white. Summing everything up and taking into account the inductive step, we end up with the following lower bound in the number of white nodes.

$$\begin{aligned}
\sum_{i=1}^{k}\frac{d^i-1}{2}+\sum_{i=k+1}^{h}\left(d^{i-k-1}\;\frac{d^{k+1}+1}{2}+1\right) &= \frac{1}{2}\sum_{i=1}^{k}d^i-\frac{k}{2}+\frac{1}{2}\sum_{i=k+1}^{h}d^i+\frac{1}{2}\sum_{i=0}^{h-k-1}d^i+h-k\\
&= \frac{1}{2}\;\frac{d^{h+1}-d}{d-1}+\frac{1}{2}\;\frac{d^{h-k}-1}{d-1}+h-\frac{3}{2}\;k\\
&= \frac{1}{2}\;\frac{d^{h+1}-1}{d-1}+\frac{d^{h-k}-1}{2\;(d-1)}+h-\frac{3}{2}\;k-\frac{1}{2}, \qquad (1)
\end{aligned}$$

which is larger than $(n+1)/2$, since $k \le h-\log_d h-1$. To see this, note that the first term in (1) is precisely $n/2$, while the function

$$f(x) \;=\; \frac{d^{h-x}-1}{2\;(d-1)}+h-\frac{3}{2}\;x-\frac{1}{2},$$

is strictly decreasing (its derivative is negative for all x when $d>1$), and

$$\begin{aligned}
f(h-\log_d h-1) &= \frac{d^{\log_d h+1}-1}{2\;(d-1)}+h-\frac{3}{2}\;\bigl(h-\log_d h-1\bigr)-\frac{1}{2}\\
&= \frac{h}{2}+\frac{h-1}{2\;(d-1)}+h-\frac{3}{2}\;h+\frac{3}{2}\;\log_d h+\frac{3}{2}-\frac{1}{2}\;>\;1.
\end{aligned}$$

Since the cardinality of neither partition set can exceed $(n+1)/2$, the previous reasoning leads to contradiction, implying that the color of the $\bigl((d^{k+1}-1)/2\bigr)$-th tree node of the $(k+1)$-st level from the right is black. Thus, at least the $(d^{k+1}-1)/2$ rightmost nodes of the $(k+1)$-st level are colored black. A similar argument about the $\bigl((d^{k+1}-1)/2\bigr)$-th tree node of the $(k+1)$-st level from the left completes the proof. ∎

In turn, Theorem 4.1 implies the following result for even integers d.

Theorem 4.2. *Let $d \ge 4$ be an even integer. There exists an n-node d-tree ($n > d$) whose optimal bisection has cutsize at least equal to*

$$\frac{d-2}{2}\Bigl(\log_d n-\log_d\log_d n-1\Bigr)-1.$$

Proof: Consider a complete $(d-1)$-tree $T^{(d-1)}$ of $n>d$ nodes, from which we derive a d-tree $T^{(d)}$ by removing the rightmost leaf l of $T^{(d-1)}$ and reattaching it as a child of the root. It is not difficult to see that the cutsize of an optimal bisection of $T^{(d)}$ is at least equal to *opt-cutsize*$(T^{(d-1)})-1$, where *opt-cutsize*$(\mathcal{T})$ denotes the cutsize of an optimal bisection of a tree $\mathcal{T}$. Indeed, assume that we have computed an optimal bisection of $T^{(d)}$. The corresponding partition sets of nodes induce a bisection of $T^{(d-1)}$, assuming that the leaf l retains its identity in both trees. Moreover, their cutsizes differ by no more than 1 ($T^{(d)}$ and $T^{(d-1)}$ differ by exactly one edge), so that the cutsize of the resulting bisection of $T^{(d-1)}$ is at most equal to $\textit{opt-cutsize}(T^{(d)}) + 1$. Therefore, $\textit{opt-cutsize}(T^{(d-1)}) \leq \textit{opt-cutsize}(T^{(d)}) + 1$, or equivalently $\textit{opt-cutsize}(T^{(d)}) \geq \textit{opt-cutsize}(T^{(d-1)}) - 1$. Then, in light of Theorem 4.1,

$$\textit{opt-cutsize}(T^{(d)}) \;\geq\; \frac{d-2}{2}\bigl(\log_{d-1} n - \log_{d-1}\log_{d-1} n - 1\bigr) - 1 \;\geq\; \frac{d-2}{2}\bigl(\log_d n - \log_d \log_d n - 1\bigr) - 1,$$

as desired. ∎

Taking into account that the cutsize is an integer, Theorems 4.1 and 4.2 can be summarized into:

Corollary 4.1. *For any integer $d \geq 3$, there exists an n-node d-tree ($n > d$) whose optimal bisection has cutsize at least equal to*

$$\left\lceil \left\lfloor \frac{d-1}{2} \right\rfloor \Bigl(\log_d n - \log_d \log_d n\Bigr) \right\rceil - \left\lfloor \frac{d}{2} \right\rfloor .$$

The combination of Corollaries 3.1 and 4.1 implies

Corollary 4.2. *For any integer $d \geq 3$, there exist n-node d-trees ($n > d$) whose optimal bisections have cutsizes equal to $\Theta(d \log_d n)$.*

Corollary 4.1 establishes the desired lower bound in the cutsize of partitions of d-trees, for $d \geq 3$. Unfortunately, in the case of 2-trees, the method used above does not apply, and a different technique is needed to produce the corresponding lower bound.

The case of 2-trees. The method that we use below to find a lower bound on the cutsize over all optimal partitions of a 2-tree is applicable without any modification to the case of a d-tree for arbitrary $d \geq 2$; it yields, however, a slightly worse lower bound $\lceil \log_d(n+1) \,/\, (2 \log_d \log_d (n+1)) \rceil$.

Let us consider a tree of n nodes; we recall from Section 2 that there exist exactly $\lfloor n/2 \rfloor$ *different* partitions of the node set into two (non-empty) sets; if two partitions both yield sets of m and $n - m$ tree nodes for some $1 \leq m \leq \lfloor n/2 \rfloor$, we say that they are *equivalent*. We also recall the correspondence between partitions and subsets of tree edges. In light of these facts, the maximum among the cutsizes of all the optimal $(m, n-m)$-partitions ($1 \leq m \leq n/2$) can be interpreted as follows: if this maximum cutsize is c, then at least one of the $\lfloor n/2 \rfloor$ possible ways to partition the tree's node set into two sets cannot be realized by any of the subsets of tree edges of sizes 1 through $c - 1$, so that we have to resort to edge subsets of size c. Our method relies precisely on this fact. For an appropriately chosen family of 2-trees, we compute an upper bound on the number of *non-equivalent* partitions that all subsets of i tree edges define, and then we estimate the minimum k for which the sum of the upper bounds for subsets of 1, 2, …, k edges is at least equal to $\lfloor n/2 \rfloor$; the estimated value of k is a lower bound on the maximum cutsize over all optimal tree partitions.

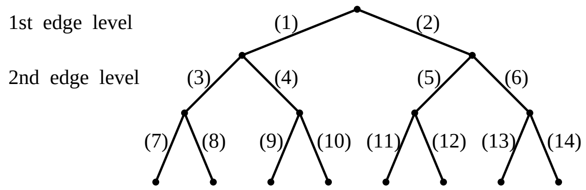

Figure 4

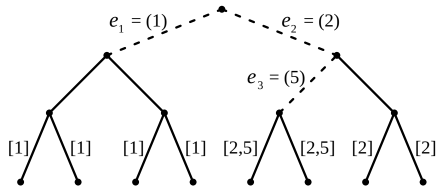

Figure 5

We consider the family of *complete 2-trees*; in particular, let T be a complete 2-tree of height $h \geq 3$ and $n = 2^{h+1} - 1$ nodes. In T, we define the notion of the i-th *edge level* $(1 \leq i \leq h)$ as the set of edges whose incident nodes have depths $i-1$ and i respectively. Additionally, in order to simplify edge identification and ordering, we associate each edge of T with a distinct integer from 1 to $n-1$: the numbers are assigned in order, as the edge levels are in turn visited from top to bottom, and the edges in each edge level are visited from left to right (Figure 4). (It is easy to prove that the 2^i edges of the i-th edge level are associated with the integers $2^i - 1$ through $2^{i+1} - 2$.)

In order to find an upper bound on the number of all non-equivalent partitions that are defined by sets of $j \geq 1$ edges of T, we construct "representative" sets R_j such that if the removal of j edges induces an $(m, n-m)$-partition of T's nodes (for some m), then there exists an element of R_j which induces an equivalent partition. This implies that the cardinality of R_j is obviously an upper bound, as desired. The sets R_j are defined in a recursive fashion: $R_0 = \{\emptyset\}$, while R_j $(j \geq 1)$ is defined in terms of R_{j-1} as follows:

$R_j = \emptyset$;
for (each set S of edges in R_{j-1}) **do**
if $(S = \emptyset)$
  then $k = 1$;
  else $k = 1 +$ maximum number associated with an edge in S;
for ($i =$ edge level of edge associated with number k; $i \leq h$; $i++$) **do**
  collect all the edges in the i-th edge level associated with numbers $\geq k$;
  label each collected edge with the edges in S that belong to the path from
    the edge to the root, or with $\emptyset$ if no such edges exist;
  partition the collected edges into classes of edges with the same label;
  for (each such class) **do**
    $e =$ the edge in the class with the smallest associated number;
    add the set $S \cup \{e\}$ to R_j;

To exemplify the different classes and labels in the last step of the above procedure, we show in Figure 5 the labels (in square brackets) of the edges of the 3rd edge level during the computation of

R_4 and when S is the set of dashed edges (associated with the numbers 1, 2, and 5). The application of the above procedure on the tree of Figure 4 to compute the sets R_1 and R_2 yields:

$$
\begin{aligned}
R_1 &= \{ \ \{(1)\},\ \{(3)\},\ \{(7)\} \ \}, \\
R_2 &= \{ \ \{(1),(2)\},\ \{(1),(3)\},\ \{(1),(5)\},\ \{(1),(7)\},\ \{(1),(11)\}, \\
&\qquad \{(3),(4)\},\ \{(3),(7)\},\ \{(3),(9)\}, \\
&\qquad \{(7),(8)\} \qquad\qquad\qquad\qquad\qquad\qquad\qquad \},
\end{aligned}
$$

where (k) denotes the edge whose assigned number is k. It is trivial to see that the three elements of R_1 correspond to three different ways to partition the 15 nodes of the tree of Figure 4 into two sets. In general, however, elements of a set R_j may define equivalent partitions. See R_2 for instance; the subsets $\{(3),(7)\}$ and $\{(7),(8)\}$ in R_2 both yield partitions of the tree's node set into sets of 2 and 13 nodes, and similarly for $\{(1),(3)\}$ and $\{(3),(9)\}$ (sets of 4 and 11 nodes), and $\{(1),(7)\}$ and $\{(3),(4)\}$ (sets of 6 and 9 nodes). We prove next that the set R_j is indeed a "representative" set for all the partitions induced by the removal of j tree edges.

Lemma 4.2. *Suppose that we apply the previously described process to generate the sets R_j on a complete 2-tree T. Then, for each set S of $k \geq 1$ edges of T, there exists an element of R_k such that the corresponding partitions are equivalent.*

Proof: We give below a constructive proof: we describe a method to transform S into an element S' of R_k such that the partitions defined by S and S' are equivalent. We start by sorting the edges in S by increasing associated number, and then we process them in order, while maintaining the invariant that processed edges are associated with numbers smaller than those of the remaining edges. Suppose that we have processed the first $i-1$ edges in S, which currently is $S = \{e_1, \ldots, e_{i-1}, d_i, \ldots, d_k\}$, the edges ordered in S by increasing associated number. Processing d_i involves: (i) each edge e that belongs to the same edge level as d_i and whose associated number is larger than that associated with e_{i-1} is labeled with the edges among $e_1, \ldots, e_{i-1}$ on the path from e to the root of T, or with $\emptyset$ if no such edges exist; (ii) next, the edges are partitioned into classes depending on their labels; (iii) finally, if e_i is the edge among those in the same class as d_i that has the smallest associated number, we substitute d_i with e_i in S, and we swap the subtrees rooted at the lowest nodes of d_i and e_i. (The labeling, classification, and selection of edges are exactly the same as the corresponding steps during the generation of R_i when the current set of edges from R_{i-1} is $\{e_1, \ldots, e_{i-1}\}$.) The swap will in general affect the d_js but none of the e_js, so that the process terminates in $|S|$ steps (Figure 6 illustrates how the above procedure transforms $S = \{(2),\ (4),\ (13)\}$ into the set $\{(1),\ (5),\ (7)\}$ that belongs to R_3; at any given moment, the set S is the set of dashed edges.) By choosing e_i among the edges in the same class as d_i and swapping the subtrees guarantees that S and $\{e_1, \ldots, e_i, d'_{i+1}, \ldots, d'_k\}$ induce equivalent partitions. By choosing the edge e_i with the smallest

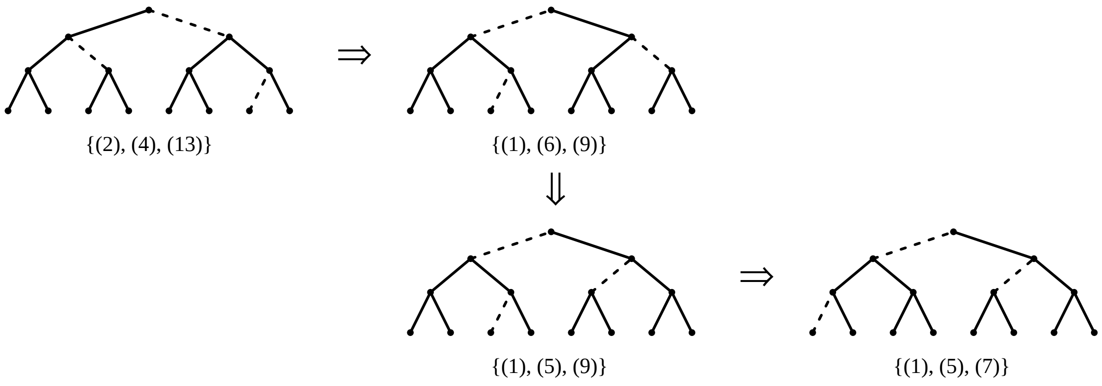

Figure 6

associated number guarantees that in the end we will get an element of R_k. Finally, we note that the number associated with e_i is larger than any of the numbers associated with the e_js but smaller than any of the numbers associated with the d'_js, thus maintaining the invariant. $\blacksquare$

Let us now estimate the cardinality $|R_j|$ of the set R_j. It is crucial to observe that in the computation of R_j the number of different classes per edge level does not exceed j. Indeed, the label of each such class is fully characterized by the labeling edge with the largest associated number; the remaining edges in the label are the edges in the path from that edge to the root. In other words, there are at most j such classes: one for each of the $j-1$ edges in S plus one for the class labeled with $\emptyset$. Since there are h edge levels, we find that

$$|R_j| \;\leq\; |R_{j-1}| \cdot h \cdot j,$$

while $|R_0| = 1$; this implies that $|R_j| \leq j!\ h^j$. Since the number of different partition ways is $\lfloor n/2 \rfloor = (n-1)/2$, the lower bound in the maximum cutsize over all optimal partitions of T's nodes is computed by finding the minimum k such that

$$\sum_{j=1}^{k} \bigl(j!\ h^j\bigr) \;\geq\; \frac{n-1}{2} \iff \sum_{j=0}^{k} \bigl(j!\ h^j\bigr) \;\geq\; \frac{n+1}{2}. \tag{2}$$

Clearly $k \geq 2$; additionally, $k \leq h$, because, for all $h \geq 2$, $h!\,h^h > 2^h = (n+1)/2$. Then, for $2 \leq k \leq h$ and $h \geq 3$, we have that

$$\sum_{j=0}^{k} \bigl(j!\ h^j\bigr) \;<\; k! \sum_{j=0}^{k} h^j \;<\; k!\ \frac{h^{k+1}}{h-1} \;<\; \frac{h^{2k}}{2} \;<\; \frac{\log_2^{2k}(n+1)}{2},$$

since $h = \log_2(n+1) - 1$. This implies that the minimum k that satisfies inequality (2) is no less than the minimum k for which

$$\frac{\log_2^{2k}(n+1)}{2} \;\geq\; \frac{n+1}{2} \iff 2^{2k\ \log_2\log_2(n+1)} \;\geq\; 2^{\log_2(n+1)} \iff k \;\geq\; \frac{\log_2(n+1)}{2\ \log_2\log_2(n+1)}.$$

Theorem 4.3. *Every complete 2-tree of $n \geq 3$ nodes admits an $(m, n-m)$-partition (for some appropriate m) whose cutsize cannot be less than*

$$\left\lceil \frac{\log_2(n+1)}{2\ \log_2 \log_2(n+1)} \right\rceil .$$

Proof: First, we note that a complete 2-tree of height 1 and 2 ($n = 3$ and 7 respectively) the above expression equals 1, which is a trivial lower bound on the cutsize of any partition. For complete 2-trees of height $h \geq 3$, the lower bound follows from the above discussion. ∎

5. Concluding Remarks.

In this paper, we proved that for $m = 1, 2, \ldots, \lfloor n/2 \rfloor$ the cutsize of an optimal $(m, n-m)$-partition of a d-tree of n nodes is $(d-1)\ \log_d m + 1$, which can be as large as $(d-1)\ \log_d n + 1$ for the case of a bisection. Additionally, we showed that for $d \geq 3$ there exist d-trees of n nodes, whose bisections admit cutsizes at least equal to $\lfloor (d-1)/2 \rfloor\ (\log_d n - \log_d \log_d n) + \Theta(d)$, thus establishing that the cutsize of a bisection of such a tree is $\Theta(d\ \log_d n)$. Unfortunately, in case $d = 2$, the corresponding lower bound is only $\lceil \log_2(n+1) \,/\, (2\ \log_2 \log_2(n+1)) \rceil$.

We mention that the cutsizes of all optimal $(m, n-m)$-partitions ($m = 1, 2, \ldots, \lfloor n/2 \rfloor$) of a tree can be computed by an algorithm based on the dynamic programming paradigm: it recursively computes the corresponding optimal cutsizes of each of the subtrees rooted at the children of the root, and then combines the results to cover all possible cases depending on whether and which among the edges incident upon the root belong to the cutset of a partition or not (see also [7]; the algorithm presented there finds only the optimal bisection, but the extension to the more general problem that we consider here is immediate). It is important to note that while combining the optimal cutsizes of the subtrees of a tree node, say, t, one does not need to explicitly go through all possible cases, that is, consider every subset E of edges among those that connect t to its children, and compute the cutsizes of partitions whose cutset contains E. Instead, we simply process each subtree T_i in turn and compute the corresponding optimal cutsizes (of the subgraph containing t, the already processed subtrees, and T_i) by taking advantage of the cutsize information associated with t (pertaining to the processed subtrees) and with T_i, and by considering two cases, namely, whether the edge connecting t to the root of T_i belongs to the cutset of a partition or not; investigating both cases ensures that all possible optimal partitions will be taken into account. From a complexity point of view, the algorithm runs in $O(n^3)$ time and requires $O(n)$ space. If, however, the cutsets of the optimal partitions need to be computed too, then the required space increases, as the cutsets of all intermediate optimal partitions need to be stored; the upper bound that we established guarantees that the needed space does not exceed $O(n\, d \log_d n)$. Finally, if the tree has a highly symmetric structure, then the algorithm may be made to run faster by precisely taking advantage of the symmetry. For instance, the time complexity of the algorithm when applied to a complete 2^i-tree (for integer i) reduces to $O(n^2)$.

We close this section by describing some open questions. First, it would be of interest to close the gap between the complexities of the upper and lower bounds for the case of 2-trees. Brief experimentation with complete 2-trees of up to 2^{21} nodes suggests that the maximum cutsize is roughly $\Theta(\log_2 n/2)$; if this is the case, we need a new method to compute a tighter lower bound. Secondly, MacGregor's upper bound for optimal bisections (see beginning of Section 3) and our corresponding lower bound (for $d \geq 3$) suggest that an upper bound for $(m, n-m)$-partitions equal to our upper bound divided by nearly 2 can probably be obtained. After all, our method to compute the upper bound collects the desired set of tree nodes only by adding subsets to it; a method where the collected set is allowed to grow larger than the desired size and subsets can be subtracted from it later may yield a smaller upper bound. Last but not least, upper and lower bounds in the cutsize of optimal partitions (and bisections in particular) of planar or more general graphs are yet to be established. Such bounds will shed new light to partitioning algorithms, such as the one in [3], whose performance depends on the cutsize of the corresponding optimal partition.

6. References.

1. R. Boppana, "Eigenvalues and graph bisection: an average-case analysis," *Proc. 28th Annual IEEE Symposium on Foundation of Computer Science* (1987), 280–285.

2. T. Bui, S. Chaudhuri, T. Leighton, and M. Sipser, "Graph bisection algorithms with good average case behavior," *Combinatorica* 7 (1987), 171–191.

3. T.N. Bui and A. Peck, "Partitioning planar graphs," *SIAM Journal on Computing* 21 (1992), 203–215.

4. M.R. Garey, D.S. Johnson, and L. Stockmeyer, "Some simplified NP-complete graph problems," *Theoretical Computer Science* 1 (1976), 237–267.

5. B.W. Kernighan and S. Lin, "An efficient heuristic procedure for partitioning graphs," *Bell System Tech. Journal* 49 (1970), 291–307.

6. S. Kirkpatrick, C.D. Gelatt, Jr., and M.P. Vecchi, "Optimization by simulated annealing," *Science* 220 (1983), 671–680.

7. R.M. MacGregor, *"On partitioning a graph: A theoretical and empirical study,"* Ph.D. Thesis, University of California, Berkeley (1978).

APPENDIX

Lemma. *The cutsize of an $(m, n-m)$-partition of the tree shown in Figure 3 (for all $m = 1, 2, \ldots, n-1$) is at most equal to $\lceil k/2 \rceil + 1$.*

Proof: It is important to observe that the structure of the tree of Figure 3 allows the collection of $(k+1)r + k$ nodes at unit cutsize, for all $r \geq 0$. We distinguish the following cases.

1. If $m \leq \lfloor k/2 \rfloor$, then the validity of the above statement obviously holds, since the cutsize of an $(m, n-m)$-partition does not exceed m.
2. If $\lfloor k/2 \rfloor + 1 \leq m \leq k$, then the desired set of m tree nodes is obtained by cutting off one of the trees marked by T_k and subsequently removing the excess $k - m \leq k - (\lfloor k/2 \rfloor + 1) = \lceil k/2 \rceil - 1$ nodes. This will result in a cutsize of at most $1 + \big(\lceil k/2 \rceil - 1\big) = \lceil k/2 \rceil$, which is less than the desired upper bound of $\lceil k/2 \rceil + 1$.
3. Finally, if $m \geq k+1$, then $m = (k+1)\pi + \upsilon$, where π is a positive integer and υ satisfies the inequalities $0 \leq \upsilon \leq k$. If $0 \leq \upsilon \leq \lfloor k/2 \rfloor - 1$, then $m = (k+1)(\pi-1) + k + (1+\upsilon)$, where a subtree of $(k+1)(\pi-1)+k$ nodes is obtained at unit cutsize, and the remaining $1+\upsilon$ nodes increase the cutsize by no more than an additional $1 + \lfloor k/2 \rfloor - 1$, so that the total cost does not exceed $1 + \lfloor k/2 \rfloor \leq \lceil k/2 \rceil + 1$. Now, if $\upsilon \geq \lfloor k/2 \rfloor$, then m can be written as $m = (k+1)\pi + k - (k-\upsilon)$, where the subtree of size $(k+1)\pi + k$ is again obtained at unit cutsize, and pruning $k - \upsilon$ nodes increases the cutsize by no more than $k - \lfloor k/2 \rfloor = \lceil k/2 \rceil$; so, the total cutsize in this case is at most $\lceil k/2 \rceil + 1$, matching the desired upper bound. ∎