J Comb Optim (2006) 12:217–230
DOI 10.1007/s10878-006-9626-z

Tree edge decomposition with an application to minimum ultrametric tree approximation

Chia-Mao Huang · Bang Ye Wu · Chang-Biau Yang

Published online: 14 August 2006

Abstract A k-decomposition of a tree is a process in which the tree is recursively partitioned into k edge-disjoint subtrees until each subtree contains only one edge. We investigated the problem how many levels it is sufficient to decompose the edges of a tree. In this paper, we show that any n-edge tree can be 2-decomposed (and 3-decomposed) within at most $\lceil 1.44 \log n \rceil$ (and $\lceil \log n \rceil$ respectively) levels. Extreme trees are given to show that the bounds are asymptotically tight. Based on the result, we designed an improved approximation algorithm for the minimum ultrametric tree.

Keywords Tree · Decomposition · Ultrametric · Approximation algorithm

1 Introduction

A partition of a tree is a collection of k edge-disjoint subtrees. Most of the previous researches about tree partitions are focus on the *vertex partition*, in which we remove $k-1$ edges and cut the tree into k subtrees. Usually the goal of partitioning a graph is to make the subgraphs as equal as possible. In this regard, different objectives may be

C.-M. Huang (✉) · C.-B. Yang
Department of Computer Science and Engineering, National Sun Yat-sen University, Kaohsiung, Taiwan 804, R.O.C.
e-mail: huangcm@par.cse.nsysu.edu.tw

C.-B. Yang
e-mail: cbyang@cse.nsysu.edu.tw

B. Y. Wu
Department of Computer Science and Information Engineering, Shu-Te University, YenChau, Kaohsiung, Taiwan 824, R.O.C.
e-mail: bangye@mail.stu.edu.tw

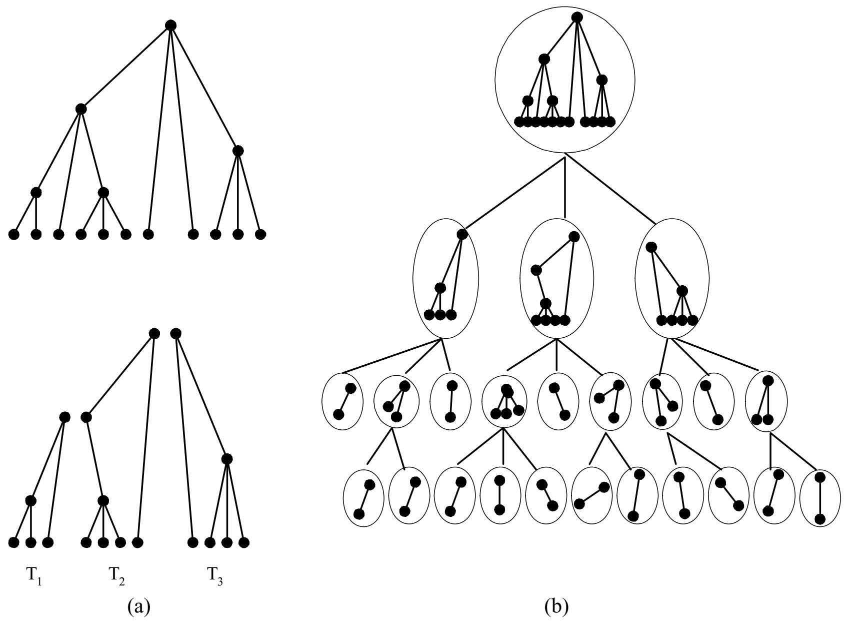

Fig. 1 (a) A 3-partition of a tree. (b) A 3-decomposition tree

defined. Algorithms for finding the optimal min-max and max-min vertex k-partitions of a tree were proposed in Becker et al. (1982) and Perl and Schach (1981), and more general objective functions had also been considered (Becker and Perl, 1983). A similar problem for partitioning a tree into minimum number of subtrees of sizes no more than a given value was investigated in Kundu and Misra (1977).

Instead of vertex partition, we consider the edge partition of a tree. An edge k-partition of tree T is a collection of k edge-disjoint subtrees whose union is the whole tree T. Figure 1(a) illustrates a feasible edge 3-partition of a tree.

A k-decomposition of a tree is a process in which the tree is k-partitioned recursively until each subtree contains only one edge. It is quite reasonable to represent the recursive process by a *decomposition tree*. A k-decomposition tree is a rooted tree and each internal node has no more than k children. In a k-decomposition tree Y of a tree T, the root represents T and each internal node represents a subtree of T. Every leaf of Y is for one edge of T. The children of an internal node represent a k-partition of the parent. An example of a 3-decomposition tree is illustrated in Fig. 1(b). The objective is to minimize the height of the decomposition tree. A related work of finding an optimal strategy for searching in a tree has been studied in Ben-Asher et al. (1999), in which the authors proposed an $O(n^4 \log^3 n)$ time algorithm for constructing a vertex decomposition tree of minimum height.

In this paper, we show that, for any tree of n edges, there exists a 2-decomposition tree with height no more than $\lceil 1.44 \log n \rceil$. Throughout this paper, $\log n = \log_2 n$. Furthermore, we show that the bound of the height is asymptotically tight by giving an extreme case. The 3-decomposition is also considered in this paper. For this case, we show that the tight bound of the height is $\lceil \log n \rceil$.

In addition to the interest in graph theory, the result of edge decomposition of a tree can be applied to the minimum ultrametric tree approximation. An ultrametric tree is a rooted, leaf-labeled, and edge-weighted tree, in which the root has the same distance to each of the leaves. An ultrametric tree is a mathematical model for the evolution of species. In an ultrametric tree, each leaf represents one species and the internal nodes represent the inferred common ancestor of the species in the subtree. More details of an ultrametric tree and its applications to computational biology can be found in Gusfield (1997).

Given a distance matrix over a set of species, the minimum ultrametric tree (MUT) problem looks for an ultrametric tree of minimum total edge length such that the distance between any pair of species is no less than the given distance. The MUT problem has been shown to be NP-hard and there is an $\varepsilon > 0$ such that the MUT problem cannot be approximated in polynomial time within ratio n^{ε} unless NP $=$ P (Farach et al., 1995). When the distance matrix is restricted to a metric, i.e. the distances satisfy the triangle inequality, the problem is also NP-hard but there exists an approximation algorithm with error ratio $1.5\lceil \log n + 1\rceil$ (Wu et al., 1999). In this paper, based on the tree edge decomposition, we improve the error ratio to $\lceil 1.44 \log n + 2\rceil$. The approximation algorithm first finds the minimum spanning tree (MST) of the input matrix. Then an edge-decomposition tree of the MST is constructed and modified to be a legal evolutionary tree topology. Finally we find the optimal weight of each tree edge with respect to the given distances and the topology.

The remaining sections are organized as follows: In Section 2, we introduce some definitions and discuss the edge partition. The 2-decomposition and the 3-decomposition are discussed in Sections 3 and 4 respectively. The application to MUT is shown in Section 5. Finally, concluding remarks are given in Section 6.

2 Preliminaries

For a tree T, let $E(T)$ denote the edge set of T, and $e(T)$ be the number of edges of T. As in the literature, the definition of *subtree* varies with whether the tree is rooted or not. A subtree of an unrooted tree is a connected induced subgraph of the tree; while, for a rooted tree, a subtree is a subgraph induced on some vertex and all its descendants.

Definition 1. An edge k-partition of a tree is a set $\{T_1, T_2, \ldots, T_k\}$ of k subgraphs of T, in which $\bigcup_i E(T_i) = E(T)$ and $E(T_i) \cap E(T_j) = \emptyset$ for all $i \neq j$.

In this paper, a null subtree in a partition is allowable.

Definition 2. Let T be a tree. An edge k-decomposition tree of T is a rooted tree defined by (i) the root represents tree T;(ii) each internal node is for a subgraph of T and each leaf is for one edge in $E(T)$; (iii) the set of the subgraphs represented by the children of an internal node is a k-partition of the tree represented by the parent.

In this paper, since we consider only edge partitions, we shall simply use the terms *partition* and *decomposition* instead of *edge partition* and *edge decomposition* respectively in the remaining paragraphs. In a rooted tree, the level of a vertex is the number of edges of the path to the root, and the height of a rooted tree is the maximum level of any vertex in the tree. In the following, we make some discussion on the edge partition of a tree. It is helpful for deriving properties of decomposition.

For any tree, there exists a vertex, known as a centroid, whose removal cutting the tree into subgraphs with nodes no more than one half of the tree. Split the branches at a centroid of a tree T into two parts as equal as possible. It can be shown that each of the two subtrees has no more than $\frac{2}{3}e(T)$ edges. The following lemma provides a more general result.

Lemma 1. *The edges of a tree T can be partitioned into two subtrees T_1 and T_2 with $\gamma \leq e(T_1) \leq 2\gamma$, for any $1 \leq \gamma \leq e(T)$. Furthermore, the partition can be found in linear time.*

Proof: Root T at an arbitrary vertex. In linear time, we can traverse the tree in postorder and compute the number of edges for the subtree rooted at each vertex. Let T_v denote the subtree rooted at v. Assume that v is a lowest vertex with $e(T_v) \geq \gamma$ and $B_1, B_2, \ldots, B_k$ are the branches at v. If $e(T_v) = \gamma$, we have done. Otherwise, we can find $j < k$ such that $\sum_{i=1}^{j-1} e(B_i) < \gamma$ and $\sum_{i=1}^{j} e(B_i) \geq \gamma$. Since $e(B_j) \leq \gamma$, we have that $\sum_{i=1}^{j} e(B_i) \leq 2\gamma$. The union $\bigcup_{i=1}^{j} B_i$ is the desired subgraph. □

Based on the above lemma, we derive the following properties for k-partition.

Corollary 1. *The edges of a tree T can be partitioned into k subtrees in such a way that each of the subtrees has no more than $\frac{2e(T)}{k+1}$ edges.*

Proof: We show the result by induction on k. Let $n = e(T)$. By taking $\gamma = n/3$ in Lemma 1, a tree T can be partitioned into two subtrees with no more than $2n/3$ edges. Therefore, the result holds for $k = 2$. For $k > 2$, by taking $\gamma = n/(k+1)$ in Lemma 1, we can partition $T = T_1 \cup Y$ with $n/(k+1) \leq e(T_1) \leq 2n/(k+1)$. By the induction hypothesis, the subtree Y can be partitioned into $k-1$ subtrees T_i, $2 \leq i \leq k$, with $e(T_i) \leq 2e(Y)/k$ for $i \geq 2$. Since $e(T_1) \geq n/(k+1)$, we have $e(Y) \leq kn/(k+1)$, and therefore $e(T_i) \leq 2n/(k+1)$. □

Similarly, the next result is based on Lemma 1 and can be shown by induction on k. Since the proof is obvious, we omit it.

Corollary 2. *The edges of a tree T can be partitioned into k subtrees in such a way that each of the subtrees has at least $\frac{e(T)}{2k-1}$ edges.*

The bounds in Corollary 1 and Corollary 2 are tight. Let T be a tree with $k+1$ branches and each of the branches contains $e(T)/(k+1)$ edges. It is obvious that, for any k-partition of T, there exists a subtree containing two of the branches. Similarly, for Corollary 2, let T be a tree with $(2k-1)$ branches and each branch has $n/(2k-1)$ edges. For any k-partition of T, there exists a subtree containing only one branch.

3 The 2-decomposition of trees

In this section, we discuss the 2-decomposition of a tree. To decompose a tree with as few levels as possible, one may want to partition the tree (and subtrees) into two parts as equal as possible in each iterations. By Corollary 1, a tree can be partitioned into two subtrees of no more than 2/3 of the edges. Consequently we have a 2-decomposition tree with height no more than $\lceil \frac{\log n}{\log(3/2)} \rceil$ for an n-edge tree. In the following, we shall present an algorithm which finds a binary decomposition tree with height no more than $\lceil \log n / \log \phi \rceil$, where $\phi = \frac{1+\sqrt{5}}{2} \approx 1.618$ is known as the *golden ratio*. We shall also show that this bound is asymptotically tight by giving an extreme tree.

At each level, instead of only considering the bipartition at the current level, we take the next level into consideration. First we find a bipartition (T_1, T') of a tree T, in which T' can be further partitioned into (T_2, T_3), and T_1, T_2 and T_3 are all small enough. At this level, T is partitioned into (T_1, T'). If (T_1, T') is good enough, the iteration is completed, and the two subtrees are decomposed recursively. Otherwise, i.e. T_1 is too small, we partition T' into (T_2, T_3) at the next level to ensure that T is partitioned into three small subtrees after two levels.

Before showing the decomposition algorithm, we show how to find the 3-partition. The next algorithm partitions a set of integers into three subsets in such a way that the sum of all elements in each subset is no more than one half of the total sum.

Algorithm partition

Input: A set $S = \{x_i | 1 \le i \le k\}$ of integers, in which $x_i \le \frac{1}{2} \sum_{x \in S} x$ for $1 \le i \le k$.
Output: A 3-partition (S_1, S_2, S_3) of S.

```
1:  $n \leftarrow \sum_{x \in S} x$.
2:  Find the maximum element $x$ in $S$.
3:  move $x$ from $S$ to $S_1$; $s_1 \leftarrow x$.
4:  for each element $x \in S$ do
        if $s_1 + x \le n/2$ then
            move $x$ from $S$ to $S_1$;
            $s_1 \leftarrow s_1 + x$.
5:  Find the maximum element $x$ in $S$.
6:  move $x$ from $S$ to $S_2$; $s_2 \leftarrow x$.
7:  for each element $x \in S$ do
        if $s_2 + x \le n/2$ then
            move $x$ from $S$ to $S_2$;
            $s_2 \leftarrow s_2 + x$.
8:  $S_3 \leftarrow S$.
```

Starting at the maximum element, the algorithm simply move every possible element into S_1 in the constraint that the total sum of the elements in S_1 is no more than $n/2$. After S_1 is completed, S_2 is built in the same way, and S_3 is the set of the remaining elements. From the algorithm, both s_1 and s_2 are no more than $n/2$. We make the following claim.

Claim 1. $\sum_{x \in S_3} x \le \frac{n}{2}$.

Proof: Let $s_3 = \sum_{x \in S_3} x$. Suppose for a contradiction that $s_3 > n/2$. There are at least two elements in S_3. Otherwise the only element in S_3 would be the unique maximum element in S, and should be moved to S_1. Therefore, there is an element $x \in S_3$ such that $x \leq \frac{1}{2}s_3$. Without loss of generality, we assume that $s_1 \geq s_2$. We have

$$\begin{aligned} x + s_2 &\leq \frac{1}{2}s_3 + s_2 \\ &= \frac{1}{2}(n - s_1 - s_2) + s_2 \\ &= \frac{1}{2}(n - s_1 + s_2) \\ &\leq \frac{1}{2}n \end{aligned}$$

However, such an element should be move to S_2 at Step 7, and this is a contradiction. □

Corollary 3. *In linear time, a tree T can be partitioned into 3 subtrees in such a way that the three subtrees share a centroid of T and each of them has no more than $\frac{e(T)}{2}$ edges.*

Proof: Root T at its centroid. Each of the branches has no more than $e(T)/2$ edges. The result follows Claim 1 and that the number of edges of every branch can be computed in linear time by traversing the tree in postorder. □

The 2-decomposition algorithm is given below, in which $1/3 \leq \lambda \leq 1/2$ is a real number to be specified later.

ALGORITHM BINARY_DECOMPOSITION

Input: A tree T.
Output: A 2-decomposition tree of T.

```
1: Find a 3-partition $\{T_1, T_2, T_3\}$ of $T$ with $e(T_i) \leq \lceil e(T)/2 \rceil$ for $1 \leq i \leq 3$,
   in which the three subtrees shares a centroid of $T$ and $T_1$ is the largest one.
2: Partition $T$ into $\{T_1, T_2 \cup T_3\}$ at this level.
3: if $e(T_1) < \lambda e(T)$ then
       Partition subtree $T_2 \cup T_3$ into $T_2$ and $T_3$ at the next level.
4: Recursively decompose each of the subtrees.
```

By Corollary 3, the 3-partition in Step 1 always exists and can be found in linear time. Note that it includes the case that T_3 is empty. In such a case, $|e(T_1) - e(T_2)| \leq 1$. In the following lemma, we show the bound of the height of the decomposition tree.

Theorem 1. *For any tree with n edges, the* BINARY_DECOMPOSITION *algorithm constructs a binary decomposition tree with height no more than $\lceil \log n / \log \phi \rceil$ in $O(n \log n)$ time.*

Proof: The algorithm recursively partitions a tree until all subgraphs contains only one edge. In each recursion, the tree is either partitioned into two subgraphs in one level or into three subgraphs in two levels. Let $n = e(T)$. In the first case, $\lambda n \leq e(T_1) \leq n/2$ and $e(T_2 \cup T_3) \leq (1-\lambda)n$. Since $(1-\lambda) \geq 1/2$, $e(T_1) \leq (1-\lambda)n$. In the other case, each of the 3 subgraphs contains no more than λn edges. Let $h(n)$ denote the height of the constructed decomposition tree of a tree with n edges. We have

$$h(n) \leq \max\{h((1-\lambda)n) + 1, h(\lambda n) + 2\} \tag{1}$$

To balance the two cases, we set the value of λ such that the following two recurrence functions are identical.

$$h_1(n) = h_1((1-\lambda)n) + 1$$
$$h_2(n) = h_2(\lambda n) + 2$$

Since $h_1(n) = h_1((1-\lambda)n) + 1 = h_1((1-\lambda)^2 n) + 2$, we have

$$(1-\lambda)^2 = \lambda \tag{2}$$

By definition, the golden ratio ϕ is a root of equation $x^2 = x + 1$. We have $\phi^2 = \phi + 1$, and

$$(1-\phi^{-2})^2 = \left(\frac{\phi^2 - 1}{\phi^2}\right)^2 = \left(\frac{1}{\phi}\right)^2 = \phi^{-2} \tag{3}$$

Equations (2) and (3) imply that $\lambda = \phi^{-2} \approx 0.382$.

Now we show that $h(n) \leq \frac{\log n}{\log \phi} \approx 1.44 \log n$ by induction. For $n = 2$, $h(2) = 1 \leq 1.44 \log 2$. Suppose that $h(i) \leq \log i / \log \phi$ for all $i < n$. Since $\lambda = \phi^{-2}$ and $\phi^2 = \phi + 1$,

$$1 - \lambda = 1 - \phi^{-2} = \frac{\phi}{\phi + 1}$$

By Eq. (1),

$$\begin{aligned} h(n) &\leq \max\{h((1-\lambda)n) + 1, h(\lambda n) + 2\} \\ &= \max\left\{h\left(\frac{\phi}{\phi+1}n\right) + 1, h\left(\frac{n}{\phi+1}\right) + 2\right\} \\ &\leq \max\left\{\frac{\log \frac{\phi n}{\phi+1}}{\log \phi} + 1, \frac{\log \frac{n}{\phi+1}}{\log \phi} + 2\right\} \\ &= \max\left\{\frac{\log \frac{\phi^2 n}{\phi+1}}{\log \phi}, \frac{\log \frac{\phi^2 n}{\phi+1}}{\log \phi}\right\} \\ &= \frac{\log n}{\log \phi} \end{aligned}$$

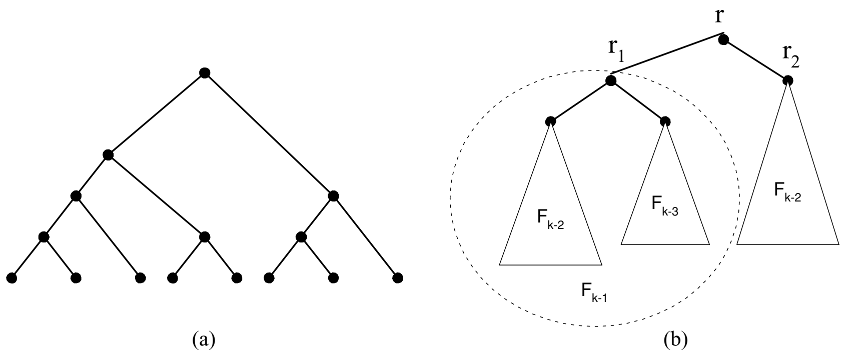

Fig. 2 Fibonacci trees. (a) A Fibonacci tree of order 6. (b) A Fibonacci tree of order k

We now turn to the time complexity. By Corollary 3, the desired 3-partition at Step 1 can be found in linear time. Since there are $O(\log n)$ levels and the time complexity for partitioning all the subgraphs of the same level is $O(n)$, the complexity of the algorithm is $O(n \log n)$. □

We are going to show that the well-known Fibonacci tree is an extreme case for the 2-decomposition, i.e. the height of any 2-decomposition tree of a Fibonacci tree is at least $1.44 \log n - C$, in which n is the number of edges and C is a constant.

Definition 3. The Fibonacci tree F_k of order k is a rooted binary tree recursively defined by (i) $F_1 = F_2$ is the tree contains only one node; (ii) for $k \geq 3$, $F_k = F_{k-1} \cup F_{k-2} \cup (r, r_1) \cup (r, r_2)$, where r is the root of F_k, and r_1 and r_2 are the roots of F_{k-1} and F_{k-2}, respectively.

Fibonacci trees of order 6 and order k are illustrated in Fig. 2. For a tree T, let $\delta_k(T)$ denote the minimum height of any k-decomposition tree of T. The next lemma shows that a Fibonacci tree F_k need at least $k-2$ levels to be decomposed.

Lemma 2. *For $k \geq 3$, $\delta_2(F_k) = k - 2$.*

Proof: We prove the lemma by induction. By definition, an unrooted F_3 is a 2-edge path, and therefore $\delta_2(F_3) = 1$. The result holds for $k = 3$. Suppose that $\delta_2(F_i) = i - 2$ for all $3 \leq i < k$. We shall show that $\delta_2(F_k) = k - 2$. By definition, $F_k = F_{k-1} \cup F_{k-2} \cup (r, r_1) \cup (r, r_2)$, in which r is the root of F_k, and r_1 and r_2 are the roots of F_{k-1} and F_{k-2}, respectively (see Fig. 2(b)). Reroot F_k at r_1 and there are three branches at r_1. For any 2-partition of an F_k, there exists a subgraph Y containing two of the three branches. That is, Y contains at least an F_{k-2}, an F_{k-3}, and two other edges. Since an F_{k-1} consists of an F_{k-2}, an F_{k-3} and a 2-edge path between their roots, we have $\delta_2(Y) \geq \delta_2(F_{k-1}) = k - 3$, and the equality holds when $Y = F_{k-1}$. Since $\delta_2(F_k) = \delta_2(Y) + 1$, we have $\delta_2(F_k) = k - 2$. □

Corollary 4. *An n-edge tree can be decomposed within* $\lceil 1.44 \log n \rceil$ *levels, and the bound is asymptotically tight.*

Proof: Theorem 1 provides the upper bound of the number of decomposition levels. We need to show that the bound is tight. By definition, the number of external nodes of a Fibonacci tree F_k is the k-th Fibonacci number f_k, where $f_1 = f_2 = 1$ and $f_k = f_{k-1} + f_{k-2}$ for $k \geq 3$. Since each internal node in F_k has exactly two children, the number of edges $e(F_k) = 2f_k - 2$. As shown in many textbooks in discrete mathematics, e.q. (Knuth, 1973, p. 82), the closed form expression for the Fibonacci numbers can be obtained by solving the recurrence relation, and f_k is the integer nearest to $\phi^k/\sqrt{5}$. Therefore we have

$$\delta_2(F_k) = k - 2 \geq \frac{\log e(F_k)}{\log \phi} \approx 1.44 \log e(F_k).$$

That is, the Fibonacci tree is an extreme example that the bound is asymptotically tight. □

4 The 3-decomposition of trees

The k-decomposition is similar to the binary decomposition except that each internal node of the decomposition tree has at most k children. In the section, we investigate the k-decomposition for $k > 2$.

By Corollary 1, a tree can be partitioned into k subtrees, each with at most a fraction $\frac{2}{k+1}$ of the total edges. By recursively applying Corollary 1, we have the next result.

Claim 2. For any $k \geq 1$, any tree of n edges can be k-decomposed with at most $\lceil \log n/(\log(k+1) - 1) \rceil$ levels.

Claim 2 provides us an upper bound of the levels of k-decomposition. However, the bounds are not tight in general. In the previous section, we have shown the tight bound $1.44 \log n$ for $k = 2$. In the following, we shall show that the tight bound for $k = 3$ is $\log n$, just the same as the one in Claim 2.

Theorem 2. *An n-edge tree can be 3-decomposed within* $\lceil \log n \rceil$ *levels, and the bound is asymptotically tight.*

Proof: By Claim 2, there is an upper bound $\log n/(\log(3+1) - 1) = \log n$. We shall show that the bound is tight by giving an extreme tree. Let B_i be a rooted complete binary tree of height i. A B_0 contains only one node by definition. Recall that $\delta_k(T)$ is the minimum level of any 3-decomposition tree of a tree T. We shall show that $\delta_3(B_i) = i$ for any $i \geq 1$ by induction.

It is obvious that $\delta_3(B_1) = 1$. Suppose that $\delta_3(B_i) = i$ for some $i \geq 1$. Let us consider the 3-decomposition of B_{i+1}. A B_{i+1} contains four B_{i-1}'s. In any 3-partition of a B_{i+1}, there is a subgraph T consisting of at least two B_{i-1}'s and two other edges. However, a B_i consists of two B_{i-1}'s and two edges. Therefore $\delta_3(T) \geq \delta_3(B_i)$.

Since $\delta_3(B_{i+1}) = \delta_3(T) + 1$, we have $\delta_3(B_{i+1}) \geq \delta_3(B_i) + 1 = i + 1$. By induction, $\delta_3(B_i) \geq i$ for any integer $i \geq 1$. Since a B_i can be obviously 3-decomposed (or 2-decomposed) with i levels, $\delta_3(B_i) = i$.

Since B_i is a complete binary tree of height i, the number of leaves is 2^i, and $e(B_i) = 2^{i+1} - 2$. Consequently $\delta_3(B_i) = \log(\frac{e(B_i)}{2} + 1) \geq \log e(B_i) - 1$, and the bound is asymptotically tight. □

5 An application to minimum ultrametric tree approximation

In this section, we propose an approximation algorithm for the minimum ultrametric tree problem on metrics (ΔMUT). Our approximation algorithm first construct a topology and then assign the edge weights. Let M be the input distance matrix and T be a given tree topology with leaf set $\{1 \ldots n\}$. It has been shown that the optimal edge weights can be easily determined in $O(n^2)$ time (Wu et al., 1999). For the completeness, we briefly explain the method as follows. For an internal node v of an ultrametric tree, let $height(v)$ be the distance from v to each leaf in the subtree. Let $d_T(i, j)$ denote the distance between i and j in a tree T. For any subgraph X of a weighted graph $G = (V, E, w)$, let $w(X)$ denotes the total edge weight of X.

By the requirement that $d_T(i, j) \geq M[i, j]$ for each pair i and j, it is obvious that $height(v) = \frac{1}{2} \max_{i,j \in T_r} M[i, j]$ for any internal node v in a minimum ultrametric tree, in which the maximum is taken over all pairs of species in the subtree rooted at v. Once $height(v)$ for each v is determined, the weight of an edge (u_1, u_2) can be assigned by $|height(u_1) - height(u_2)|$. Another useful property in the previous work is that the total edge weight of an ultrametric tree can be computed as follows. For a binary ultrametric tree T with root r,

$$w(T) = height(r) + \sum_{v \in V(T)} height(v). \tag{4}$$

The difference between our algorithm and the previous one (Wu et al., 1999) is that the topology we used is an edge decomposition tree of a minimum spanning tree (MST) of the input matrix, in which a MST of a matrix means a MST of the complete graph corresponding to the matrix.

An evolutionary tree is a decomposition of the set of species. If we label each internal node by the set of its descendants (leaves in the subtree), the label of any internal node is partitioned into the two labels of its children. An edge decomposition tree is not a valid evolutionary tree since it decomposes a tree into edges but not vertices, and some modifications are needed.

By Theorem 1, there exists an edge decomposition tree with at most $\lceil \log_\phi n \rceil$ levels for a tree of n edges. Let Y be such an edge decomposition tree of the MST T of M. For a graph G, let $V(G)$ denote its vertex set. First we define two labels (functions) on the $V(Y)$.

- For any node $v \in V(Y)$, the subgraph of T represented by v is denoted by $SG(v)$.
- Let $SS : V(Y) \rightarrow 2^{V(T)}$, which labels each internal node of Y by a subset of $V(T)$ according to the following rules. For the root r of Y, $SS(r) = V(T)$. Let $v \in V(Y)$

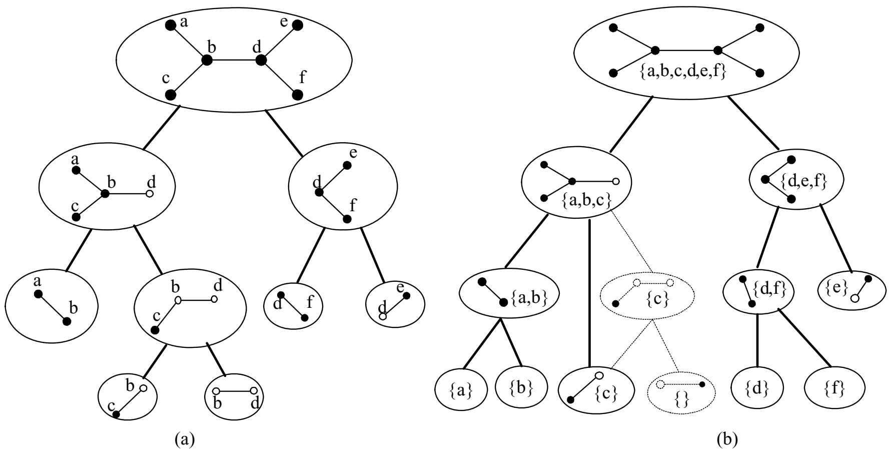

Fig. 3 (a) An edge decomposition tree of a tree and $SG(v)$ for each node v. (b) The modified edge-decomposition tree and $SS(v)$ for each vertex v. The dotted vertices are discarded

and v_1, v_2 be the two children of v. By definition, $\{SG(v_1), SG(v_2)\}$ is an edge partition of $SG(v)$, and therefore $SG(v_1)$ and $SG(v_2)$ share one common vertex u. Define $SS(v_1) = V(SG(v_1)) \cap SS(v)$ and $SS(v_2) = (V(SG(v_2)) - \{u\}) \cap SS(v)$.

It should be noted that $SS(v_1)$ and $SS(v_2)$ form a set partition of $SS(v)$. For an edge decomposition tree, the labeling SS is not unique. Particularly, the common node u is arbitrarily assigned to $SS(v_1)$ or $SS(v_2)$ if $u \in SS(v)$. It is also possible that $SS(v) = \emptyset$ for some $v \in V(Y)$. To use Y as the topology of our ultrametric tree, we make the following modifications.

- For any leaf v of Y, $SG(v)$ is an edge of T, and therefore $SS(v)$ contains at most two vertices. For each vertex v, if $SG(v)$ is an edge and $SS(v) = \{i, j\}$, we create two child vertices v_1 and v_2 for it, and let $SS(v_1) = \{i\}$, $SS(v_2) = \{j\}$, and $SG(v_1) = SG(v_2) = \emptyset$.
- Any vertex v with $SS(v) = \emptyset$ is discarded.
- Any vertex v with only one child is also discarded and its child is directly connected to its parent.

An example of an edge decomposition tree and its modification are illustrated inFig. 3. The modified edge decomposition tree is a valid evolutionary tree and has the following properties.

P1: The number of levels in Y is at most $\lceil \log_{\phi} n + 1 \rceil$.

P2: Let v be an internal node of Y and have children v_1 and v_2. We have $w(SG(v_1)) + w(SG(v_2)) = w(SG(v))$ since $SG(v_1)$ and $SG(v_2)$ form a partition of $SG(v)$. Let $LV(i)$ denote the set of nodes which are at the i-th level of Y. Recursively applying the property, we have

$$\sum_{v \in LV(i)} w(SG(v)) \le w(SG(r)),$$

where r is the root of Y.

P3: For each node $v \in V(Y)$, $\max\{M[i, j] | i, j \in SS(v)\} \leq w(SG(v))$ since the distance between two species is no more than the distance on the tree $SG(v)$.

Our approximation algorithm for the ΔMUT is listed below.

ALGORITHM APP-ULTRAMETRIC

Input: An $n \times n$ metric M.
Output: An ultrametric tree T.

```
1: Find the minimum spanning tree $MST(M)$ of $M$.
2: Find a modified edge decomposition tree $Y$ of $MST(M)$, of which the number
   levels is no more than $\lceil \log_\phi n + 1 \rceil$.
3: Compute $height(v)$ for each internal node $v$ of $Y$ and assign the optimal
   edge weight.
```

To analyze the approximation ratio, we use a previous result for the lower bound of the optimal. In the next lemma, *TSP*(M) denotes the minimum length among all Hamiltonian cycles of the graph corresponding to the metric M.

Lemma 3. *Let $T^* = (V, E, w)$ is the minimum ultrametric tree for a metric M. $w(T^*) \geq \frac{1}{2}TSP(M)$ (Wu et al., 1999).*

Since *TSP*(M) is obviously no less than $w(MST(M))$, we have the following corollary.

Corollary 5. *The total edge weight of the minimum ultrametric tree for a metric M is no less than $\frac{1}{2}w(MST(M))$.*

We now show the performance of the approximation algorithm in the next theorem.

Theorem 3. *Given an $n \times n$ metric, the* APP-ULTRAMETRIC *algorithm builds an ultrametric tree T with approximation ratio $\lceil \log_\phi n + 2 \rceil$ in $O(n^2)$ time, where $\phi = \frac{\sqrt{5}+1}{2}$.*

Proof: First we show the time complexity. Step 1 takes $O(n^2)$ time since the minimum spanning tree of a complete graph can be found in $O(n^2)$ time. By Theorem 1 and the previous work, Steps 2 and 3 can be done in $O(n \log n)$ and $O(n^2)$ time respectively. The time complexity of the algorithm is therefore $O(n^2)$.

We now turn to the approximation ratio. The topology Y of the ultrametric tree T constructed by the algorithm is a modified edge decomposition tree of the MST of G. The edge weights are assigned in the way that *height*(v) is one half of the maximum distance $M[i, j]$ between any species i and j in the subtree. Then, by Property P3, *height*$(v) \leq \frac{1}{2}w(SG(v))$, and we have

$$\sum_{v \in V(T)} height(v) \leq \frac{1}{2} \sum_{v \in V(T)} w(SG(v)) \tag{5}$$

Let r denote the root of T. By Property P2, the sum of $w(SG(v))$ of all vertices at the same level is no more than $w(SG(r))$. Since $SG(r)$ is $MST(M)$, we have

$$\begin{aligned}\sum_{v\in V(T)} w(SG(v)) &= \sum_{i}\sum_{v\in LV(i)} w(SG(v))\\ &\leq \sum_{i} w(SG(r)) = \sum_{i} w(MST(M))\end{aligned}\tag{6}$$

By Property P1, the number of levels in Y is at most $\lceil \log_\phi n + 1\rceil$ except the leaves. Combining Eqs. (5), (6) and Property P1, we have

$$\sum_{v\in V(T)} height(v) \leq \frac{1}{2}\left\lceil \log_\phi n + 1\right\rceil w(MST(M))\tag{7}$$

By Eqs. (4) and (7),

$$\begin{aligned}w(T) &= height(r) + \sum_{v\in V(T)} height(v)\\ &\leq \frac{1}{2}\left\lceil \log_\phi n + 2\right\rceil w(MST(M))\end{aligned}$$

By Corollary 5, since the $\frac{1}{2}w(MST(M))$ is a lower bound of the total edge weight of the minimum ultrametric tree, the approximation ratio is $\left\lceil \log_\phi n + 2\right\rceil \approx 1.44\log n + 2$. $\square$

6 Concluding remarks

In this paper, we show the tight bounds of the levels of the 2-decomposition and 3-decomposition of a tree. The result for the 2-decomposition is applied to approximating the minimum ultrametric tree. An open question is the tight bound of the k-decomposition for $k > 3$.

We have shown that a complete binary tree is an extreme case for 3-decomposition. The result can be easily extended to $k > 3$. Similar to the proof of Theorem 2, it can be shown that a complete k-ary tree of height i need i levels to be $(k + 1)$-decomposed. It provides us a lower bound $\log n/\log(k - 1)$ for the level of k-decomposition. However the lower bounds meets the upper bounds in Claim 2 only for $k = 3$.

References

Becker R, Perl Y (1983) Shifting algorithms for tree partitioning with general weighting functions. J Algorithms 4:101–120

Becker R, Schach SR, Perl Y (1982) A shifting algorithm for Min-Max tree partitioning. J ACM 29:58–67

Ben-Asher Y, Farchi E, Newman I (1999) Optimal search in trees. SIAM J Comput 28:2090–2102

Farach M, Kannan S, Warnow T (1995) A robust model for finding optimal evolutionary trees. Algorithmica 13:155–179

Gusfield D (1997) Algorithms on strings, trees, and sequences—Computer science and computational biology, Cambridge University Press

Knuth DE (1973) The art of computer programming; Vol. 3 Sorting and Searching. Addison-Wesley, Reading, Massachusetts

Kundu S, Misra J (1977) A linear tree partitioning algorithm. SIAM J Comput 6:151–154

Perl Y, Schach SR (1981) Max-Min tree partitioning. J ACM 28:5–15

Wu BY, Chao KM, Tang CY (1999) Approximation and exact algorithms for constructing minimum ultrametric trees from distance matrices. J Comb Optim 3:199–211