On the uniform edge-partition of a tree

Bang Ye Wu [a,1] Hung-Lung Wang [b] Shih Ta Kuan [a]
Kun-Mao Chao [b,c]

[a] *Dept. of Computer Science and Information Engineering, Shu-Te University, YenChau, KaoShiung, Taiwan 824, R.O.C.*

[b] *Dept. of Computer Science and Information Engineering, National Taiwan University, Taipei, Taiwan 106, R.O.C.*

[c] *Graduate Institute of Networking and Multimedia, National Taiwan University, Taipei, Taiwan 106, R.O.C.*

Abstract

We study the problem of uniformly partitioning the edge set of a tree with n edges into k connected components, where $k \leq n$. The objective is to minimize the ratio of the maximum to the minimum number of edges of the subgraphs in the partition. We show that, for any tree and $k \leq 4$, there exists a k-split with ratio at most two. For general k, we propose a simple algorithm that finds a k-split with ratio at most three in $O(n \log k)$ time. Experimental results on random trees are also shown.

Key words: tree, partition, optimization problem, algorithm.

1 Introduction

Graph partition is an important problem in computer science. It finds applications in parallel computing, data storage and segmentation, and operation research. Most of the previous research was devoted to the *vertex partition*,

Email addresses: bangye@mail.stu.edu.tw (Bang Ye Wu),
r92085@csie.ntu.edu.tw (Hung-Lung Wang),
s91113111@mail.student.stu.edu.tw (Shih Ta Kuan),
kmchao@csie.ntu.edu.tw (Kun-Mao Chao).
[1] Corresponding author

and many variants of the problem have been defined and investigated with different objectives and constraints. To measure how uniform a partition is, three natural objectives are usually used.

- To minimize the maximum (min-max).
- To maximize the minimum (max-min).
- To minimize the ratio of the maximum to the minimum (min-ratio).

Many problems in this line of investigation have been shown to be NP-hard [1,10]. For the vertex partition of a tree, polynomial time algorithms for both the min-max and the max-min objectives were developed [5,7,15,17]. Becker and Perl [6] summarized their previous results with some other co-authors and showed that the tree vertex partition problem of several other objective functions can also be solved by using the shifting algorithm. An open problem in that paper is the *most uniform vertex partitioning* problem for trees, in which the objective is to minimize the difference between the maximum and the minimum weights of the vertex set in the partition. For a special case that the tree is a path, a solution was given in [16]. One can image that the problem is more difficult than the min-max or max-min problem since both the smallest and the largest parts are concerned.

In this paper, we study the problem of splitting a tree into k parts with approximately equal number of *edges* in each part subject to that the edges in each part are connected. How well can one do it?

More formally, we define a k-split of a tree T as follows. Let T be a tree and $1 \leq k \leq e(T)$. A k-tuple $(T_1, T_2, \ldots, T_k)$ is a k-split of T if (1) each T_i is a

connected subgraph of T; and (2) T_i and T_j are edge disjoint for $i \neq j$; and (3) the union of all the subgraphs forms the whole tree T.

For partitioning the tree into two parts, i.e. 2-split, all the three objectives are equivalent. When the number of parts is larger than two, their worst cases might differ from each other. Although the worst cases of the k-splits of a tree for both min-max and max-min objectives can be easily shown (see Corollaries 5 and 6 in Section 2), it is much more involved for the min-ratio problem. We show that, in the algorithmic aspect, to find an optimal k-split of a tree with respect to each of the three objectives is NP-hard, even for unweighted trees. We focus on the worst case analysis of the ratio, and prove that, for any tree and $k \leq 4$, there exists a k-split with ratio at most two. For general k, we propose a simple algorithm that finds a k-split with ratio at most three in $O(n \log k)$ time. Experimental results on random trees are also shown.

The study on edge partition is helpful for the multiserver routing problem on a tree [2,3]. In such a problem, we are given a tree and k identical servers, and ask for a route for each server such that each vertex is visited by at least one server. An edge partition of the tree is a feasible solution of that problem, and a "fair" partition balances the loads (the routing distances) of the servers.

Another application of the tree edge partition is for the multiple sequence alignment (MSA) problem, which is important in computational biology. One way to get an alignment is to employ a tree, in which each vertex represents one sequence and each edge corresponds to an alignment of two sequences [4,11,18,20]. More details about MSA and algorithms for constructing such a

guide tree are referred to [12,19]. For n sequences, each with length m, the time complexity of such an approach is $O(nm^2)$ and is very time consuming for large n and m. Since one tree edge corresponds to performing a pairwise alignment, a k-split of the tree partitions the whole work into k parts and derives a parallel algorithm for the problem. To balance the working load, a k-split with small ratio should be applied.

The rest of the paper is organized as follows. In Section 2, we define some notations, explain the computational complexity of the problem, and show some preliminary results. The worst cases of the min-ratio for $k = 3$ and $k = 4$ are discussed in Section 3. In Section 4, we show a simple algorithm for general k with ratio at most three, and present some experimental results. Finally. concluding remarks are given in Section 5.

2 Notations and Preliminaries

Let $E(T)$ denote the edge set of a tree T and $e(T)$ denote the number of edges of tree T. Throughout this paper, $n = e(T)$. An edge with endpoints u and v is denoted by (u, v). Let T be a rooted tree and v be a vertex of T. We use T_v to denote the subtree rooted at v, i.e. the subgraph induced on v and all its descendants. Let u be a child of v. The subgraph $T_u \cup (u, v)$ is called a *branch* of v.

Definition 1: Let T be a tree and $1 \leq k \leq e(T)$. The ratio of a k-split $(T_1, T_2, \ldots, T_k)$ of T is defined by $\frac{\max_i\{e(T_i)\}}{\min_i\{e(T_i)\}}$.

By $T = A \uplus B$, we denote that T is split into A and B, i.e., the edge sets of the two subgraphs form a partition of $E(T)$. It is also noted that A and B share a common vertex if $T = A \uplus B$. By $T = A \uplus B \uplus C$, we understand a 3-split (A, B, C) of T, in which B intersects with both A and C. It includes the case that the three subgraphs share a common vertex.

PROBLEM: Minimum Ratio k-Split

INSTANCE: A tree T and an integer $1 < k < e(T)$.

GOAL: Find a k-split of T with minimum ratio.

The min-max and the max-min k-split problem are defined similarly except that the objectives are to minimize the maximum subgraph, and to maximize the minimum subgraph respectively. We can easily show that all the three problems are NP-hard by a simple reduction from the following problem.

PROBLEM: 3-Partition

INSTANCE: A bound $B \in Z^+$ and a set A of $3m$ integers a_i, $1 \leq i \leq 3m$, satisfying $B/4 < a_i < B/2$ and $\sum_i a_i = mB$.

QUESTION: Can A be partitioned into m disjoint sets A_i, $1 \leq i \leq m$, such that $\sum_{a \in A_i} a = B$ for $1 \leq i \leq m$ (Note that each A_i must therefore contain exactly three elements from A.)?

Given A and B as an instance of the 3-partition problem, we construct a tree T consisting of a root r and $3m$ branches Y_i, $1 \leq i \leq 3m$, incident with the root, in which Y_i is an arbitrary tree of a_i edges. It is easy to see that there exists an m-split of T with ratio one if and only if the answer of the 3-partition problem is "yes". Since the 3-partition problem is NP-complete in the strong

sense [10], we have the following result.

Theorem 1: The Minimum Ratio k-Split problem is NP-hard.

Obviously, the reduction remains true for the min-max and the max-min objective functions.

Corollary 2: Both the Min-Max and the Max-Min k-Split problems are NP-hard.

The next lemma appeared in [13]. For convenience, we rewrite it and give a proof for the completeness.

Lemma 3: Let T be a rooted tree. For any $1 \leq \gamma \leq e(T)$, we can split T into (T_1, T_2) at a vertex v in linear time such that $\gamma \leq e(T_1) \leq 2\gamma$, in which v is a vertex satisfying $e(T_v) \geq \gamma$ and $e(T_u) < \gamma$ for any child u of v.

Proof: In linear time, we can traverse the tree in the post order and compute the number of edges for the subtree rooted at each vertex. Such a vertex v can be easily found while traversing the tree. Assume that $B_1, B_2, \ldots, B_k$ are the branches at v. If $e(T_v) = \gamma$, we are done. Otherwise, we can find $j \leq k$ 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. $\square$

To show that the bounds are tight, consider a tree consisting of exact three branches, each with $n/3$ edges, incident with the centroid, where a centroid of a tree is a vertex u such that while rooting at u, no branch of u contains more than one half of the vertices. Taking $\gamma = n/3$ in Lemma 3, we have the following result.

Corollary 4: For any tree T, there is a 2-split of T with ratio at most two. The numbers of the two subgraphs are at most $2n/3$ and at least $n/3$. Furthermore, such a 2-split can be found in $O(n)$ time and the bounds are tight.

Corollary 4 shows the worst case of $k = 2$ for the min-max, the max-min, and the min-ratio objectives. For the min-max and the max-min objectives, we can easily extend it to k-split for $k > 2$ as follows.

Corollary 5: For any tree T and $k \geq 2$, there is a k-split of T such that each subtree has at most $\frac{2n}{k+1}$ edges, and the bound is tight.

Proof: We show the result by induction. Given a tree T, by Lemma 3, we find $T = T_1 \uplus T'$ such that

$$\frac{n}{k+1} \leq e(T_1) \leq \frac{2n}{k+1}.$$

Suppose by induction hypothesis that T' can be split into $k-1$ subgraphs, each with at most $2e(T')/k$ edges. Since $e(T_1) \geq n/(k+1)$, the number of edges of each subgraph is upper bounded by

$$\frac{2(n - n/(k+1))}{k} = \frac{2n}{k+1}.$$

The tightness of the bound can be easily shown by considering an extreme case in which the tree has $k+1$ branches at the root and each has exactly $n/(k+1)$ edges. $\square$

Corollary 6: For any tree T and $k \geq 2$, there is a k-split of T such that each subtree has at least $\frac{n}{2k-1}$ edges, and the bound is tight.

Proof: Similarly, we show the result by induction. Given a tree T, by Lemma 3, we find $T = T_1 \uplus T'$ such that

$$\frac{n}{2k-1} \leq e(T_1) \leq \frac{2n}{2k-1}.$$

Suppose by induction hypothesis that T' can be split into $k-1$ subgraphs, each with at least $e(T')/(2k-3)$ edges. Since $e(T_1) \leq 2n/(2k-1)$, the number of edges of each subgraph is lower bounded by

$$\left(n - \frac{2n}{(2k-1)}\right) \frac{1}{2k-3} = \frac{n}{2k-1}.$$

The tightness of the bound can be easily shown by considering an extreme case in which the tree has $2k-1$ branches at the root and each has exactly $n/(2k-1)$ edges. For this instance, at least one subtree contains only one of the branches. $\square$

Consider the extreme case for the min-max objective given in the proof of Corollary 5, i.e., a tree T with $k+1$ branches at the root and each has exactly $n/(k+1)$ edges. In any k-spilt of T, at least one of the subtrees contains only one branch and there must be a subtree containing at least two of the branches. In other words, the min-ratio is two. This instance shows that the worst min-ratio for any tree is lower bounded by two. To be a worst case, it is required to show that any tree can be split with ratio at most two. However, we did not find a simple proof as in the min-max and max-min cases.

The following simple result shows an upper and a lower bounds for the sizes of the subgraphs in a k-split with a limited ratio.

Lemma 7: If $(T_1, T_2, \ldots, T_k)$ is a k-split of T with ratio r, then, for each subgraph T_i, $\frac{n}{r(k-1)+1} \leq e(T_i) \leq \frac{rn}{k+r-1}$.

Proof: Let x be the number of edges of the maximum component. Since the number of edges of the minimum component is no more than the mean of the remainder, i.e., $\frac{n-x}{k-1}$,

$$x \leq \frac{r(n-x)}{k-1}.$$

Solving the inequality, we have $x \leq \frac{rn}{k+r-1}$. Similarly, let y denote the minimum number of edges. The maximum is no less than the mean of the remainder, $\frac{n-y}{k-1}$, and we have $y \geq \frac{(n-y)}{r(k-1)}$, which implies $y \geq \frac{n}{r(k-1)+1}$. $\square$

Particularly, for $r = 2$, the number of edges of each subgraph in a k-split with ratio at most two is between $\frac{n}{2k-1}$ and $\frac{2n}{k+1}$. In the next lemma, we show that there is a similar result for the min-ratio but with a stronger condition, and this result will be used as one of the cases for proving the bound of the min-ratio.

Lemma 8: Let $T = Y \uplus T'$ and $(T_1, T_2, \ldots, T_{k-1})$ be a $(k-1)$-split of T' with ratio at most 2. If $\frac{n}{k+1} \leq e(Y) \leq \frac{2n}{2k-1}$, then $(Y, T_1, T_2, \ldots, T_{k-1})$ is a k-split of T with ratio at most two.

Proof: Let $t_{\max} = \max_i e(T_i)$, $t_{\min} = \min_i e(T_i)$, and $y = e(Y)$. It is sufficient to show that $\frac{t_{\max}}{2} \leq y \leq 2t_{\min}$. By Lemma 7, $t_{\max} \leq 2(n-y)/k$ and $t_{\min} \geq (n-y)/(2k-3)$. Since $y \geq n/(k+1)$, we have

$$\frac{t_{\max}}{y} \leq \frac{2(n-y)}{ky} = \frac{2n}{ky} - \frac{2}{k} \leq \frac{2(k+1)}{k} - \frac{2}{k} = 2.$$

Similarly, since $y \leq 2n/(2k-1)$,

$$\frac{t_{\min}}{y} \geq \frac{(n-y)}{y(2k-3)} = \frac{n}{y(2k-3)} - \frac{1}{2k-3} \geq \frac{2k-1}{2(2k-3)} - \frac{1}{2k-3} = 1/2.$$

□

3 Worst cases of 3-splits and 4-splits

Now let us consider the 3-split of a tree. By Lemma 8, if a tree T can be split into $Y \uplus T'$ such that $\frac{e(T)}{k+1} \leq e(Y) \leq \frac{2e(T)}{2k-1}$, we can find a 3-split of T with ratio at most two. If it is not the case, we show in the following that a 3-split with ratio at most two always exists for any tree. First, we establish a 3-split which will be used as a basis of our discussion for $k = 3$ and 4. In the remaining paragraphs, we shall use the following notations: Let $x = e(X)$, $y = e(Y)$, $x_i = e(X_i)$, and $y_i = e(Y_i)$ for $i = 1, 2$.

Claim 9: For any $k \geq 3$, a tree T can be split into $X \uplus P \uplus Y$ such that $\frac{n}{k+1} \leq x, y \leq \frac{2n}{k+1}$.

Proof: Root T at an arbitrary vertex. By Lemma 3, we split $T = X \uplus T_1$ at a vertex u such that $\frac{n}{k+1} \leq e(X) \leq \frac{2n}{k+1}$. Then, root T_1 at u, and we can split another subgraph Y, $\frac{n}{k+1} \leq e(Y) \leq \frac{2n}{k+1}$, from T_1 at a vertex v. Note that u and v are not necessarily distinct. □

Claim 10: Let $3 \leq k \leq 4$ and X be a tree rooted at u and $\frac{2n}{2k-1} \leq x \leq \frac{2n}{k+1}$. If each branch at u has no more than $\frac{n}{k+1}$ edges, X can be split into X_1 and X_2 at u such that $x_1 \geq x_2$ and $\frac{n}{2k-1} \leq x_1 \leq \frac{2n}{2k-1}$.

Proof: First we show that a subgraph X_1 can be split from X at u such that $\frac{n}{2k-1} \leq e(X_1) \leq \frac{2n}{2k-1}$. If there exists a branch of more than or equal

to $n/(2k-1)$ edges, the branch is the desired subgraph since $n/(k+1) < 2n/(2k-1)$. Otherwise, the result directly follows Lemma 3.

Second, we show that we can assume that $x_1 \geq x_2$ without loss of the generality. Suppose that $x_1 < x_2$. Since, for $k \leq 5$,

$$x_2 = x - x_1 < \frac{2n}{k+1} - \frac{n}{2k-1} < \frac{2n}{2k-1},$$

the number of edges of X_2 is also in the desired range, and we may exchange X_1 and X_2. $\square$

Theorem 11: For any tree T, a 3-split of T with ratio at most two can be found in $O(n)$ time.

Proof: By Claim 9, we can find $T = X \uplus P_0 \uplus Y$ such that

$$\frac{n}{4} \leq y \leq x \leq \frac{n}{2}.$$

We consider the following two cases.

- Case 1: $y \leq 2n/5$.

  In this case T can be split into $Y \uplus T_1$ such that $n/4 \leq y \leq 2n/5$. By Corollary 4, there is a 2-split (P_1, P_2) of T_1 with ratio at most two. By Lemma 8, (Y, P_1, P_2) is a 3-split of T with ratio at most two.

- Case 2: $2n/5 < y \leq x \leq n/2$.

  As in Claim 10, we split $X = X_1 \uplus X_2$ such that $n/5 \leq x_1 \leq 2n/5$ and $x_1 \geq x_2$, in which $x_1 = e(X_1)$ and $x_2 = e(X_2)$. It should be noted that $X_2 \cup P_0$ is connected since X_1 and X_2 are split at the vertex shared by X and P_0. If $x_1 \geq n/4$, it is similar to Case 1. Otherwise we have $n/5 \leq x_1 < n/4$. Since

$x_1 \geq x/2$ and $x \geq y$, it follows that $x_1 \geq y/2$. By $e(X_2 \cup P_0) = n - x_1 - y$, we have

$$n/4 < e(X_2 \cup P_0) < 2n/5.$$

Consequently, $(X_1, X_2 \cup P_0, Y)$ is a 3-split of T with ratio at most two.

We have shown that there exists a 3-split with ratio at most 2 in each of the two cases, and the proof is completed since the time complexity is obviously $O(n)$. $\square$

Next, we turn to the 4-splits. We show the following result.

Theorem 12: For any tree T, there exists a 4-split of T with ratio at most two.

Proof: The discussion is divided into 6 cases. For each case, we show how to obtain a 4-split with ratio at most two. For the details, please refer to Appendix A. $\square$

The next corollary is directly from the above theorem.

Corollary 13: Given a tree T of n edges, a 4-split of T with ratio at most two can be found in $O(n)$ time.

4 On general k

4.1 A simple algorithm

We now propose a simple algorithm which finds a k-split of a tree with ratio at most three. Given a tree T and an integer k, the algorithm starts at the 1-split (T) and repeatedly computes a $(i+1)$-split from the i-split by 2-splitting the maximum subgraph. The time complexity of this algorithm is $O(n \log k)$.

Algorithm Simple-Split

Input: A tree T and an integer $k \leq e(T)$.

Output: A k-split of T.

1: Initiate an empty queue Q of trees, and insert T into Q.

2: For $i \leftarrow 1$ to $k-1$ do

2.1: Choose a tree Y in Q with maximum number of edges.

2.2: Find a 2-split (Y_1, Y_2) of Y with ratio at most two.

2.3: Remove Y from Q.

2.4: Insert Y_1 and Y_2 into Q.

3: Output the k trees in Q as the k-split of T.

In the next theorem, we show the performance of the algorithm.

Theorem 14: Given a tree T with n edges and an integer $k \leq n$, the algorithm **Simple-Split** finds a k-split of T with ratio at most 3 in $O(n \log k)$ time.

Proof: Let M_i and m_i be respectively the maximum and minimum numbers of edges of trees in the queue Q at i-th iteration. We first claim that the ratio M_i/m_i is at most 3 for each i. Initially Q contains only the input tree T, and $M_1/m_1 = 1$. Suppose that $M_i/m_i \leq 3$ for some i. We shall show that $M_{i+1}/m_{i+1} \leq 3$, and then the above claim is consequently true by induction. At $(i+1)$-th iteration, the maximum tree Y is chosen and split into Y_1 and Y_2 with ratio at most 2. Therefore, $M_{i+1} \leq M_i$, and $m_{i+1} = \min\{m_i, e(Y_1), e(Y_2)\}$. Since $\min\{e(Y_1), e(Y_2)\} \geq e(Y)/3 = M_i/3$ and $M_i/m_i \leq 3$, we have

$$\frac{M_{i+1}}{m_{i+1}} \leq \frac{M_i}{M_i/3} = 3.$$

Next, we turn to the time complexity. Let $f_n(i)$ be the total time complexity of executing Step 2.2 in the first i iterations. By Corollary 4, splitting a tree of M_i edges at i-th iteration takes $O(M_i)$ time. Since the ratio M_i/m_i is at most three, by Lemma 7, we have

$$M_i \leq \frac{3n}{i+2}.$$

Therefore, for some constant c, $f_n(1) \leq cn$, and

$$f_n(i) \leq f_n(i-1) + c\frac{3n}{i+2}$$

for $i > 1$. Solving the recurrence relation, we have

$$\begin{aligned} f_n(k) &\leq c\sum_{i=1}^{k} \frac{3n}{i+2} \\ &< 3cn\sum_{i=1}^{k} \frac{1}{i} = 3cnH_k, \end{aligned}$$

in which H_k is the well-known k-th *harmonic number*. Since $H_k = O(\log n)$, we obtain $f_n(k) = O(n \log k)$.

For Step 2.1, 2.3, and 2.4, by simply using a data structure like *heap* to store the numbers of edges of the trees in the queue, all the operations can be done in totally $O(k \log k)$ time. Therefore the total time complexity is $O(n \log k)$. □

4.2 Experimental results on random trees

To investigate the practical behavior of the algorithm **Simple-Split**, we implement the algorithm and perform some tests on random trees. Before showing the experimental results, we explain how we find the 2-split at Step 2.2. By Corollary 4, we can find a 2-split of ratio at most two by the procedure described in the proof of Lemma 3. However, although it ensures the bound of the worst ratio, the procedure does not try to find the best one. We use the following procedure to find a 2-split. For a given tree Y, we root the tree at its centroid. Initially we regard each branch as a subgraph, and then repeatedly merge the smallest two subgraphs until only two subgraphs are left. To see that the procedure always returns a 2-split (Y_1, Y_2) of ratio at most two, it is sufficient to show that the smaller subgraph Y_1 contains at least $e(Y)/3$ edges. Since the tree is rooted at its centroid, each branch contains no more than $e(Y)/2$ edges. If $e(Y_1) < e(Y)/3$, it implies that $e(Y_2) < 2e(Y)/3$ since Y_2 is either a single branch or obtained by merging two subgraphs smaller than Y_1. But $e(Y_1) + e(Y_2) = e(Y)$, and it is a contradiction.

For different n (number of edges) and k, we recorded the ratios of the k-splits found by the algorithm. Since the program runs very fast, we do not show the

execution time. For each (n, k), hundreds of instances were tested, and the average ratios are shown in Table 1.

Table 1
The average ratios

n	100	500	1000	3000	6000	10000
$k = 2$	1.21	1.23	1.20	1.17	1.19	1.21
$k = 3$	1.85	1.84	1.85	1.87	1.89	1.85
$k = 4$	1.51	1.50	1.46	1.39	1.45	1.48
$k = 5$	1.97	1.99	2.00	2.06	1.98	1.97
$k = 10$	2.09	2.11	2.08	2.13	2.09	2.10
$k = 20$	2.23	2.26	2.21	2.22	2.17	2.17
$k = 50$	3.00	2.38	2.41	2.41	2.39	2.43
$k = 100$	1.00	2.43	2.50	2.54	2.51	2.50

Since the worst cases (ratio 3) do exist, showing the worst ratios in the test is meaningless. The more instances we run, the larger the worst ratio is. Instead, we show the distributions for some typical pairs (n, k). In Table 2, we show the percentage of the ratio in each specified range. For example, the value 65.3 in the cell of the second row and third column means that, for $n = 100$ and $k = 4$, there are 65.3% of the instances in the test such that the ratio of the obtained split is less than or equal to 1.6.

By the experimental results, we observed the following.

- For small k, the algorithm performs well, but the obtained ratios get larger and tend toward the worst case as k increasing. Observing the cases of $k = 2$, we find that the algorithm splits a tree into two parts quite evenly, and it is also the reason why the performance is good for $k = 4$ but rather bad for $k = 3$.

Table 2
The distribution of ratios (in percentage)

(n,k)	≤ 1.4	≤ 1.6	≤ 1.8	≤ 2	≤ 2.2	≤ 2.4	≤ 2.6	≤ 2.8
$(100,4)$	45.9	65.3	83.2	93.9	98.4	99.2	99.8	100
$(500,4)$	48.0	73.2	88.2	96.7	98.5	99.4	100	100
$(100,10)$	0.4	1.7	9.9	48.5	72.0	87.5	97.4	99.8
$(500,10)$	0	1.7	9.0	40.2	68.4	86.5	95.7	99.5
$(5000,10)$	0.2	1.5	9.3	35.4	68.3	86.4	95.4	99.3
$(500,20)$	0	0	2.1	20.8	50.6	76.4	92.3	98.7
$(5000,20)$	0	0	0.6	15.1	49.6	76.0	92.1	99.1

- As long as k is small with respect to n, the results are almost not affected by n. In Table 1, we can see that the average ratios in each row are almost the same except for $(n,k) = (100,50)$ and $(100,100)$. For these two cases, k is so large (with respect to n) that the results are obvious and somewhat meaningless.
- The distributions approximate to the normal distribution. For each (n,k), the standard deviation is approximately 0.27. In our test, the obtained ratios of about 70% of the instances are in the range $[\mu-\sigma, \mu+\sigma]$, in which μ is the mean and σ is the standard deviation.
- There are many instances that the algorithm obtained a ratio larger than two. In this aspect, it is significant to develop an algorithm always finding a ratio within two for general k. Even for $k=3$ and $k=4$, there are still about respectively 35% and 5% of the instances in our test such that the obtained ratios are larger than two. Therefore, the results for 3-splits and 4-splits in this paper are useful in some contexts.

5 Concluding Remarks

One of the most important open problems in this line of investigation is that whether there exists a k-split with ratio at most two for general k. Our future work includes exact and approximation algorithms for finding the min-ratio k-split for general or fixed k.

Acknowledgments

We thank the referees for their helpful comments. Bang Ye Wu was supported in part by an NSC grant 93-2213-E-366-014 from the National Science Council, Taiwan. Hung-Lung Wang and Kun-Mao Chao were supported in part by an NSC grant 94-2213-E-002-091 from the National Science Council, Taiwan.

References

[1] G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccamela and M. Protasi, *Complexity and Approximation — Combinatorial optimization problems and their approximability properties*, Springer Verlag, 1999.

[2] I. Averbakh and O. Berman, A heuristic with worst-case analysis for minimax routing of two traveling salesmen on a tree, *Discrete Appl. Math.*, 68:17–32, 1996.

[3] I. Averbakh and O. Berman, $(p-1)/(p+1)$-approximate algorithms for p-traveling salesmen problem on a tree with minmax objective, *Discrete Appl. Math.*, 75:201–216, 1997.

[4] V. Bafna, E. L. Lawler and P. Pevzner, Approximation algorithms for multiple sequence alignment, *Theoretical Computer Science*, 182:233–244, 1997.

[5] R. Becker and Y. Perl, Shifting algorithms for tree partitioning with general weighting functions, *J. Algorithms*, 4:101–120, 1983.

[6] R. Becker and Y. Perl, The shifting algorithm technique for the partitioning of trees, *Discrete Appl. Math.*, 62:15–34, 1995.

[7] R. Becker, S.R. Schach and Y. Perl, A shifting algorithm for min-max tree partitioning, *J. ACM*, 29:58–67, 1982.

[8] R. Becker, B. Simeone and Y.-I Chiang, A shifting algorithm for continuous tree partitioning, *Theor. Comput. Sci.*, 282:353–380, 2002.

[9] G.J. Chang and F.K. Hwang, Optimality of consecutive and nested tree partitions, *Networks*, 30:75–80, 1997.

[10] M.R. Garey and D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W.H. Freeman and Company, San Francisco, 1979.

[11] D. Gusfield, Efficient methods for multiple sequence alignment with guaranteed error bounds, *Bulletin of Mathematical Biology*, 55:141–154, 1993.

[12] D. Gusfield, *Algorithms on Strings, Trees, and Sequences – Computer Science and Computational Biology*, Cambridge University Press, 1997.

[13] C.M. Huang, B.Y. Wu and C.B. Yang, Tree edge decomposition with an application to minimum ultrametric tree approximation, unpublished manuscript.

[14] I. Krasikov, On a tree cutting problem of P. Ash, *Discrete Math.*, 93:51–61, 1991.

[15] S. Kundu and J. Misra, A linear tree partitioning algorithm, *SIAM J. Comput*, 6:151–154, 1977.

[16] M. Lucertini, Y. Perl and B. Simeone, Most uniform path partitioning and its use in image processing, *Discrete Appl. Math.* 42:227-256, 1993.

[17] Y. Perl and S.R. Schach, Max-min tree partitioning, *J. ACM*, 28:5–15, 1981.

[18] P. Pevzner, Multiple alignment, communication cost, and graph matching, *SIAM J. Appl. Math.*, 52:1763–1779, 1992.

[19] B.Y. Wu and K.-M. Chao, *Spanning Trees and Optimization Problems*, Chapman & Hall / CRC Press, 2004.

[20] B.Y. Wu, G. Lancia, V. Bafna, K.-M. Chao, R. Ravi and C.Y. Tang, A polynomial time approximation scheme for minimum routing cost spanning trees, *SIAM J. Comput.*, 29:761–778, 2000.

Appendix A: Proof of Theorem 12.

Similar to the proof of Theorem 11, we start at splitting T into $X \uplus P_0 \uplus Y$ as in Claim 9 such that

$$\frac{n}{5} \leq y \leq x \leq \frac{2n}{5}.$$

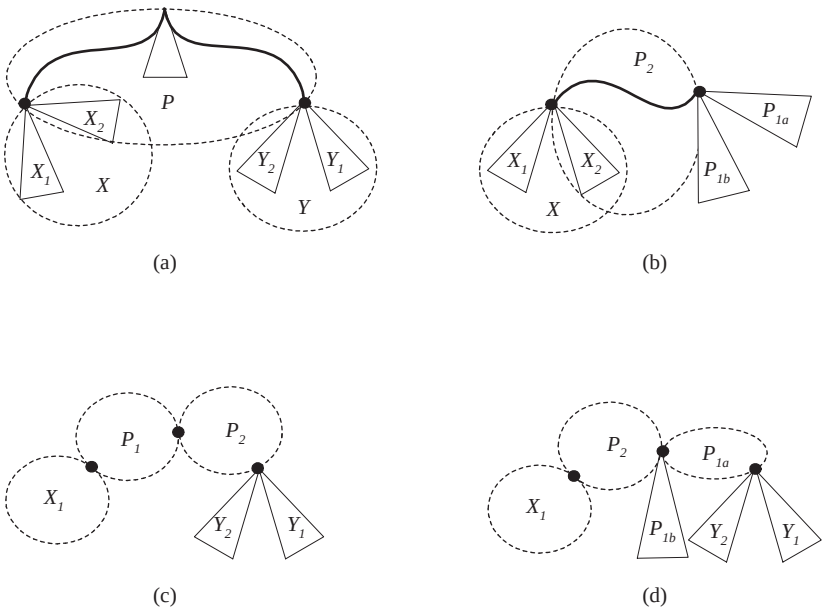

Fig. 1. 4-split cases

Case 1: $y \leq 2n/7$.

In this case T can be split into $Y \uplus T_1$ such that $n/5 \leq y \leq 2n/7$. By Theorem 11, there is a 3-split (P_1, P_2, P_3) of T_1 with ratio at most two. By Lemma 8, (Y, P_1, P_2, P_3) is a 4-split of T with ratio at most two.

Case 2: $2n/7 < y \leq x \leq 2n/5$.

By Claim 10, we split $X = X_1 \uplus X_2$ such that $x_1 \geq x_2$ and $n/7 \leq x_1 \leq 2n/7$. If $x_1 \geq n/5$, it is similar to Case 1, and therefore we assume that $n/7 \leq x_1 < n/5$. Similarly we split $Y = Y_1 \uplus Y_2$ such that $y_1 \geq y_2$ and $n/7 \leq y_1 < n/5$. Let $P = P_0 \cup X_2$, and we have $T = X_1 \uplus P \uplus Y$ as in Figure 1.(a). Remember that $2x_1 \geq y$.

By the property of a centroid, P can be split into three subgraphs P_2, P_{1a} and P_{1b} (possibly null) at its centroid in such a way that each of the subgraphs has no more than $\lceil e(P)/2 \rceil$ edges. If there are only two branches and $e(P)$ is an odd number, we add a dummy edge incident with the centroid to simplify

the proof. One may check that the correctness is not affected. Therefore we can assume that each of the three subgraphs has no more than $e(P)/2$ edges. Let P_2 be the largest and $P_1 = P_{1a} \cup P_{1b}$. We have

$$e(P_2) \leq e(P_1) \leq 2e(P_2). \tag{1}$$

Since $x_1 < n/5$ and $y \leq 2n/5$, we have

$$e(P_1) > n/5 > x_1. \tag{2}$$

Since $x_1 \geq n/7$ and $y \geq 2n/7$, we have

$$e(P_2) \leq \frac{1}{2}(n - x_1 - y) \leq \frac{2n}{7} \leq y. \tag{3}$$

By Eqs. (1)–(3) and $x_1 \leq y \leq 2x_1$, we further divide this case into the following subcases:

- $y/2 \leq e(P_2) \leq e(P_1) \leq 2x_1$.
- $e(P_1) > 2x_1$.
- $e(P_2) < y/2$.

For each case, we shall show that there exists a desired 4-split.

- Case 2.1: $y/2 \leq e(P_2) \leq e(P_1) \leq 2x_1$. In this case (X_1, Y, P_1, P_2) is a desired 4-split.
- Case 2.2: $e(P_1) > 2x_1$. we divide into two subcases.

  Case 2.2.1: P_1 adjacent to X_1. Let $Q = P_1 \cup X_1$. Split Q into Q_1 and Q_2 such that

  $$\frac{e(Q)}{3} \leq e(Q_2) \leq e(Q_1) \leq \frac{2e(Q)}{3}. \tag{4}$$

We show that (P_2, Y, Q_1, Q_2) is a desired 4-split. First, since $e(P_2) \geq e(P_1)/2 > x_1 \geq y/2$, we have

$$e(P_2) \leq y \leq 2e(P_2) \tag{5}$$

Since $e(P_1) > 2x_1$,

$$e(Q_2) \geq \frac{1}{3}(e(P_1) + x_1) > x_1 \geq y/2. \tag{6}$$

Since $x_1 < e(P_2)$ and $e(P_1) \leq 2e(P_2)$,

$$e(Q_1) \leq \frac{2}{3}(e(P_1) + x_1) < 2e(P_2). \tag{7}$$

By Eqs. (4)–(7), the result follows.

Case 2.2.2: P_2 adjacent to X_1. In this case, P_2 contains X_2 (Figure 1.(b)) since $e(P_2) > x_1 \geq x_2$. Let $P_{2a} = P_2 - X_2$ and $e(P_{1a}) \geq e(P_{1b})$. We show that $(X, P_{2a} \cup P_{1b}, P_{1a}, Y)$ is a desired 4-split.

Since $e(P_2) \geq e(P_{1a})$, $e(P_1) > 2x_1$, and $x_1 \geq x_2$, we have

$$\begin{aligned} e(P_{2a}) + e(P_{1b}) &= e(P_2) + e(P_{1b}) - x_2 \\ &\geq e(P_1) - x_2 > 2x_1 - x_2 \geq \frac{x}{2}. \end{aligned} \tag{8}$$

Since $e(P_{1a}) \geq e(P_{1b})$ and $e(P_1) > 2x_1$,

$$e(P_{1a}) > x_1 \geq \frac{x}{2}. \tag{9}$$

Since $x + y \geq 4n/7$ and $x \geq y$, we have that $x \geq 2n/7$ and that

$$e(P_{2a} \cup P_{1b}) + e(P_{1a}) = n - (x + y) \leq 3n/7.$$

By Eq. (8), $e(P_{2a} \cup P_{1b}) \geq x/2 \geq n/7$ and therefore $e(P_{1a}) \leq 2n/7 \leq x$. Similarly $e(P_{2a} \cup P_{1b}) \leq x$. That is, X is the largest subgraph. Since all the three smaller subgraphs has at least $x/2$ edges, the ratio is at most two.

- Case 2.3: $e(P_2) < y/2$. We divide into two subcases.

Case 2.3.1: P_2 adjacent to Y (Figure 1.(c)). We show that $(X_1, P_1, P_2 \cup Y_2, Y_1)$ is a desired 4-split. Since $y_2 \leq y_1$ and $e(P_2) < y/2 \leq y_1$, we have

$$e(P_2 \cup Y_2) \leq 2y_1. \tag{10}$$

Since $e(P_2) \geq (n - x_1 - y)/3$,

$$\begin{aligned} e(P_2) + y_2 &\geq (n - x_1 - y_1)/3 \\ &\geq (n - n/5 - n/5)/3 \\ &= n/5 \geq y_1 \end{aligned}$$

Combined with Eq. (10), we have

$$y_1 \leq e(P_2 \cup Y_2) \leq 2y_1. \tag{11}$$

By Eq. (2) and $e(P_1) \leq 2e(P_2) < y \leq 2x_1$, we have

$$x_1 \leq e(P_1) \leq 2x_1, \tag{12}$$

and

$$e(P_1) < y \leq 2y_1. \tag{13}$$

Since $e(P_2) < y/2$ and $y_2 \leq y/2$,

$$e(P_2 \cup Y_2) < y \leq 2x_1. \tag{14}$$

By Eqs. (11)–(14), the result follows.

Case 2.3.2: P_1 adjacent to Y (Figure 1.(d)). Suppose that Y is adjacent to P_{1a}, and here P_{1a} may be larger or smaller than P_{1b}. In this case, we show that $(X_1, P_2 \cup P_{1b}, P_{1a} \cup Y_2, Y_1)$ is a desired 4-split.

Since $y_2 \leq y_1$ and $e(P_{1a}) < e(P_2) < y/2 \leq y_1$, we have

$$e(P_{1a} \cup Y_2) \leq 2y_1. \tag{15}$$

Similarly,

$$e(P_{1a} \cup Y_2) \leq 2x_1. \tag{16}$$

Since $e(P_2) < y/2$ and $e(P_{1b}) \leq e(P_2)$,

$$e(P_2 \cup P_{1b}) < y \leq 2y_1. \tag{17}$$

Similarly,

$$e(P_2 \cup P_{1b}) < 2x_1. \tag{18}$$

Since $e(P_2 \cup P_{1b}) < 2y_1$, $y_1 < n/5$, and $x_1 < n/5$, we have

$$e(P_{1a} \cup Y_2) \geq n - (x_1 + y_1 + e(P_2 \cup P_{1b})) > n/5. \tag{19}$$

Similarly,

$$e(P_2 \cup P_{1b}) > n/5. \tag{20}$$

By Eqs. (19) and (20), in $(X_1, P_2 \cup P_{1b}, P_{1a} \cup Y_2, Y_1)$, the subgraph X_1 or Y_1 is the smallest and no less than a half of the maximum (by Eqs. (15) – (18)). Therefore it is a 4-split with ratio at most two.