SIAM J. DISCRETE MATH.
Vol. 21, No. 2, pp. 532–547
© 2007 Society for Industrial and Applied Mathematics

OPTIMAL TREE STRUCTURES FOR GROUP KEY MANAGEMENT WITH BATCH UPDATES*

RONALD L. GRAHAM†, MINMING LI‡, AND FRANCES F. YAO‡

Abstract. We investigate the key management problem for broadcasting applications. Previous work showed that batch rekeying can be more cost-effective than individual rekeying. Under the assumption that every user has probability p of being replaced by a new user during a batch rekeying period, we study the structure of the optimal key trees. Constant bounds on both the branching degree and the subtree size at any internal node are established for the optimal tree. These limits are then utilized to give an $O(n)$ dynamic programming algorithm for constructing the optimal tree for n users and any fixed value of p. In particular, we show that when $p > 1 - 3^{-1/3} \approx 0.307$, the optimal tree is an n-star, and when $p \leq 1 - 3^{-1/3}$, each nonroot internal node has a branching degree of at most 4. We also study the case when p tends to 0 and show that the optimal tree resembles a balanced ternary tree to varying degrees depending on certain number-theoretical properties of n.

Key words. key trees, group keys, batch updates, optimality

AMS subject classifications. 05C05, 49K99

DOI. 10.1137/06064929X

1. Introduction.

In the group broadcast problem, we have n subscribers and a group controller (GC) that periodically broadcasts messages (e.g., a video clip) to all the subscribers over an insecure channel. To guarantee that only the authorized users can decode the contents of the messages, the GC will dynamically update the group key for the whole group. Whenever some user leaves or joins, the GC will generate a new group key and in some way notify the remaining users in the group. A recent survey of the key management problem for groups of low-state devices can be found in [1]. Here, we consider the key tree model [4] for the key management problem. We describe this model briefly as follows (precise formulation is given in section 2). Every leaf node of the key tree represents a user and stores his individual key. Every internal node stores a key shared by all leaf descendants of the internal node. Every user possesses all the keys along the path from the leaf node (representing the user) to the root node. To prevent revoked users from knowing future message contents and also to prevent new users from knowing past message contents, the GC updates a set of keys, whenever a new user joins or a current user leaves, as follows. So long as there is a user change among the leaf descendants of an internal node v, the GC will: (1) replace the old key stored at v with a new key, and (2) broadcast (to all users) the new key encrypted with the key stored at each child node of v. Note that only users represented by leaf descendants of v can get useful information from the broadcast. Furthermore, this procedure must be done in a bottom-up fashion (i.e., starting with

*Received by the editors January 6, 2006; accepted for publication (in revised form) January 8, 2007; published electronically June 28, 2007. The work described in this paper was supported in part by the National Natural Science Foundation of China (60135010, 60321002, 60223004), the Chinese National Key Foundation Research & Development Plan (2004CB318108), the U.S. National Science Foundation grant CCR-0310991, and a grant from the Research Grants Council of Hong Kong under project CityU 1165/04E.

http://www.siam.org/journals/sidma/21-2/64929.html

†Department of Computer Science and Engineering, University of California at San Diego, La Jolla, CA 92037 (graham@ucsd.edu).

‡Department of Computer Science, City University of Hong Kong, Kowloon, Hong Kong, People's Republic of China (minmli@cs.cityu.edu.hk, csfyao@cityu.edu.hk).

the lowest v whose key must be updated—see section 2 for details) to guarantee that a revoked user will not know the new keys. The cost of the above procedure counts the number of encryptions used in step 2 (or equivalently, the number of broadcasts made by the GC).

When users change frequently, this method for updating the group keys whenever a user leaves or joins may be too costly. Thus, a batch rekeying strategy was proposed by Li et al. in [2] whereby rekeying was done only periodically instead of immediately after each member change. They designed a marking algorithm for processing the batch updates. It was shown by simulation that, using their algorithm, among the totally balanced key trees (where internal nodes of the tree have branching degree 2^i), a degree of 4 is the best when the number of requests (leave/join) within a batch is not large. For a large number of requests, using a star (a tree of depth 1) to organize the users outperforms all the balanced key trees mentioned above in their simulation. Further analysis of the batch rekeying model was done by Zhu, Chan, and Noubir in [5]. They introduced a new model to investigate the special case when the number of joins always equals the number of leaves during a batch period. Thus, they assumed that during a certain period, every user has a probability p of being replaced by a new user. They then studied the optimal tree under two assumptions: (A) the tree is totally balanced, and (B) every node on level i has 2^{k_i} children. Under these assumptions, characterizations of the optimal key tree were given together with an algorithm for computing it. In particular, it was shown that every node on an intermediate level of the tree should have a degree of 4 (that is, $i = 2$).

In this paper, we carry out the first theoretical investigation for the optimal tree as modelled by [5] but without assumptions (A) and (B). In the following discussion, an "optimal tree" always refers to a true minimum-cost key tree without any a priori restrictions on its structure; i.e., we allow the degree of each node in the tree to be arbitrary and independent of each other. Denote such a tree by a (p, n)-*optimal tree* where n is the number of users (i.e., number of leaves of the tree), and p is the probability that a leaf gets updated. The main results of the paper are as follows. We identify for p a range $1 \geq p \geq 1 - 3^{-1/3} \approx 0.307$, where a star (a tree of depth 1) is the (p, n)-optimal tree for all n. We also prove, for all (p, n)-optimal trees, a constant upper bound 4 to the branching degree of any node v other than the root, and an upper bound (as a function of p) to the size of the subtree rooted at v. These characterizations enable us to design a dynamic programming algorithm to compute the optimal tree in time $O(n)$. We further study the case when $p \to 0$ and show that the optimal tree is as close to a balanced ternary tree with n leaves as possible, subject to number-theoretical properties of n (see Theorem 4 for a precise statement).

The rest of the paper is organized as follows. In section 2, we describe the batch update model in detail. In section 3, we find the range of p when the (p, n)-optimal tree is a star. We derive properties of the general optimal trees in section 4 and give a linear-time dynamic programming algorithm for constructing them in section 5. Finally, we carry out the analysis for the case $p \to 0$ in section 6 and show that, in most cases, a branching degree of 3 is employed by the optimal tree.

2. Preliminaries. Before giving a precise formulation of the tree optimization problem to be considered, we briefly discuss its motivation and review the basic key tree model for group key management. This model is referred to in the literature either as key tree [4] or LKH (logical key hierarchy) [3].

In the key tree model, there is a GC, represented by the root, and n subscribers (or users) represented by the n leaves of the tree. The tree structure is used by the

GC for key management purposes. Associated with every node of the tree (whether internal node or leaf) is an encryption key. The key associated with the root is called the traffic encryption key (TEK), which is used for accessing encrypted service contents by the subscribers. The key k_v associated with each nonroot node v is called a key encryption key (KEK) which is used for updating the TEK when necessary. Each subscriber possesses all the keys along the path from the leaf representing the subscriber to the root.

In the batch update model to be considered, only simultaneous join/leave is allowed, that is, whenever there is a revoked user, a new user will be assigned to that vacant position. This assumption is justified since, in a steady state, the number of joins and departures would be roughly equal during a batch processing period. To guarantee forward and backward security, a new user assigned to a position (leaf) will be given a new key by the GC and, furthermore, all the keys associated with the ancestors of the leaf must be updated by the GC. The updates are performed from the lowest ancestor upward for security reasons. We will explain the updating procedure together with the updating cost in what follows.

The GC first communicates with each new subscriber separately to assign a new key to the corresponding leaf. After that, the GC will broadcast certain encrypted messages to all subscribers in such a way so that each valid subscriber will know all the new keys associated with its leaf-to-root path while the revoked subscribers will not know any of the new keys. The GC accomplishes this task by broadcasting the new keys, in encrypted form, from the lowest level upward recursively as follows. Let v be an internal node at the lowest level whose key needs to be (but has not yet been) updated. For each child u of v, the GC broadcasts a message containing $E_{k_u^{new}}(k_v^{new})$, which means the encryption of k_v^{new} with the key k_u^{new}. Thus the GC sends out d_v broadcast messages for updating k_v if v has d_v children. Updating this way ensures that the revoked subscribers will not know any information about the new keys while current subscribers can use one of their KEKs to decrypt the useful $E_{k_u^{new}}(k_v^{new})$ sequentially until they get the new TEK.

We adopt the probabilistic model introduced in [5] that each of the n positions has the same probability p to independently experience subscriber change during a batch rekeying period. Under this model, an internal node v with N_v leaf descendants will have a probability of $1 - q^{N_v}$ for which its associated key k_v requires updating, where $q = 1 - p$. The updating incurs $d_v \cdot (1 - q^{N_v})$ expected broadcast messages by the procedure described above. We thus define the expected updating cost $C(T)$ of a key tree T by $C(T) = \sum_v d_v \cdot (1 - q^{N_v})$, where the sum is taken over all the internal nodes v of T. It is more convenient to remove the factor d_v from the formula by associating the cost $1 - q^{N_v}$ with each of v's children. This way we express $C(T)$ as a node weight summation: for each nonroot tree node u, its node weight is defined to be $1 - q^{N_v}$, where v is u's parent. The optimization problem we are interested in can now be formulated as follows.

Optimal key tree for batch updates. We are given two parameters $0 \le p \le 1$ and $n > 0$. Let $q = 1 - p$. For a rooted tree T with n leaves and node set V (including internal nodes and leaves), define a weight function $c(u)$ on V as follows. Let $c(r) = 0$ for root r. For every nonroot node u, let $c(u) = 1 - q^{N_v}$, where v is u's parent. Define the cost of T as $C(T) = \sum_{u \in V} c(u)$. Find a T for which $C(T)$ is minimized. We say that such a tree is (p, n)-*optimal* and denote its cost by $OPT(p, n)$.

We will first study the case when p is any fixed constant and later the case for $p \to 0$. Notice that $C(T) \ge \sum_v d_v(1 - q) \ge pn$ implies $OPT(p, n) \ge pn$. On

the other hand, we have $OPT(p,n) \le (1-q^n)n$ by considering a tree where all leaves are attached directly to the root (i.e., a star). Thus we know asymptotically $OPT(p,n) = \Omega(n)$ when p is a constant. However, it is still interesting to identify the *exact* optimal tree which can achieve significantly better cost than a star, especially when p is a small constant.

3. The star optimal bound. We start with some basic definitions about rooted trees. We say a tree is of *depth* k if the longest leaf-root path consists of k edges. A tree of depth 2 is also referred to as a *two-level tree*. A tree of depth 1 is called a *k-star* if it has k leaves. A tree edge (u,v), where u is a child of v, is said to be at *depth* k if the path from u to the root consists of k edges. The *branching degree* of a node v is the number of children of v; the *subtree size* of v, denoted by N_v, refers to the number of leaf descendants of v.

LEMMA 1. *If the* n-star *can achieve* $OPT(p,n)$*, then the* $(n-1)$-star *can also achieve* $OPT(p,n-1)$.

Proof. We prove the lemma by contradiction. Suppose the $(n-1)$-star cannot achieve $OPT(p,n-1)$. Let the degree of the root in a $(p,n-1)$-optimal tree be k, where $k < n-1$. Write the optimal cost as $OPT(p,n-1) = k(1-q^{n-1}) + C$, where C represents the contribution to the cost by edges at depth ≥ 2. Thus we have $(n-1)(1-q^{n-1}) > k(1-q^{n-1}) + C$, which implies $n(1-q^n) > (k+1)(1-q^n) + C$. This means we can reduce the cost of $OPT(p,n)$ by adopting the same structure of the $(p,n-1)$-optimal tree but with root degree $k+1$, a contradiction. ∎

LEMMA 2. *When* $0 \le q \le 3^{-1/3}$*, the* n-star *is strictly better than any two-level tree.*

Proof. A two-level tree can be obtained from a star by successively grouping certain nodes together to form a subtree of the root. To prove the lemma, we need only show that the above operation always increases the cost of the tree; i.e., for any grouping size k, where $1 < k < n$, we will show that $1-q^n < \frac{1}{k}(1-q^n) + 1 - q^k$. This is trivially true when $k = 1$ or $q = 0$, so we assume that $k \ge 2$ and $q > 0$.

With fixed $q > 0$, define $f(k)$ for integer k by $f(k) = k\log_k(1/q)$. We observe that for any fixed q, where $0 < q < 1$, the value of $f(k)$ is minimized when $k = 3$. Thus, when $0 < q \le 3^{-1/3}$, we have $k\log_k(1/q) \ge 1$ which implies $kq^k \le 1$. Hence, for $0 < q \le 3^{-1/3}$, we have

$$\begin{aligned} 1 - q^n - \left(\frac{1}{k}(1-q^n) + 1 - q^k\right) &= \frac{1}{k}(kq^k - 1 - (k-1)q^n) \\ &< \frac{1}{k}(kq^k - 1) \\ &\le 0. \end{aligned}$$ ∎

LEMMA 3. *Let* p *and* n *be given. If the* n-star *is strictly better than any two-level tree, then the* n-star *is the* (p,n)*-optimal tree.*

Proof. If the n-star is not a (p,n)-optimal tree, then we can transform the (p,n)-optimal tree from the bottom up, every time combining two levels into a star. By Lemma 1, the m-star is strictly better than any two-level tree for $1 < m < n$ and p. Since one level is always better than two levels, we can eventually transform the optimal tree into the n-star without increasing the cost. ∎

THEOREM 1. *When* $1 \ge p \ge 1 - 3^{-1/3}$*, the* n-star *is the* (p,n)*-optimal tree for any* n*. For* $2 \le n \le 4$*, the* n-star *is the* (p,n)*-optimal tree for any* $p > 0$.

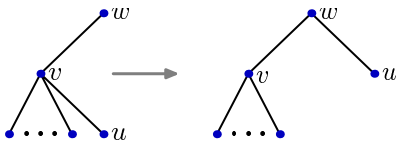

FIG. 1. *Tree transformation* 1.

Proof. The first part of the theorem follows from Lemmas 2 and 3. The cases of $2 \le n \le 4$ can be verified easily. □

4. Properties of an optimal tree. By Theorem 1, the structure of a (p, n)-optimal tree is uniquely determined for $0 \le q \le 3^{-1/3} \approx 0.693$ (or $1 \ge p \ge 1-3^{-1/3} \approx 0.307$). We now derive some properties of the optimal trees which will be used for constructing a (p, n)-optimal tree in the remaining range $1 \ge q > 3^{-1/3}$. Note that Lemmas 4 and 5 as well as Theorem 2 are true for all (p, n)-optimal trees where $0 \le p \le 1$ and $n > 0$.

For a tree T, we associate a value $t_v = q^{N_v}$ with every node v (thus $t_v = q$ if v is a leaf). The subtree rooted at u is denoted by T_u. We say T_u is a subtree of v if u is a child of v.

LEMMA 4. *For a nonroot internal node v with a branching degree of k in a (p, n)-optimal tree, every child u of v satisfies $t_u \ge \frac{k-1}{k}$.*

Proof. Assume $t_u < \frac{k-1}{k}$, we can then move u up to become a sibling of v, as shown in Figure 1. In this way, we increase the total cost of the tree by

$$\begin{aligned}\Delta C &= (1 - q^{N_w}) + (k-1)(1 - q^{N_v - N_u}) - k(1 - q^{N_v}) \\ &< 1 + (k-1)(1 - q^{N_v - N_u}) - k(1 - q^{N_v - N_u} t_u) \\ &= q^{N_v - N_u}(kt_u - (k-1)) \\ &< 0,\end{aligned}$$

where w is the parent of v and N_v represents the value before the transformation. This contradicts the cost optimality of the original tree. □

LEMMA 5. *Every nonroot internal node in a (p, n)-optimal tree has a branching degree of ≤ 5.*

Proof. By Lemma 4, if a nonroot internal node v in the optimal tree has a branching degree of $k \ge 6$, then for any child u of v we have $t_u \ge \frac{k-1}{k}$. We can group together two children of v, with the largest and the second largest t_u values, to form a single subtree of v as shown in Figure 2. By this transformation, we increase the total cost by

$$\begin{aligned}\Delta C &= 2(1 - t_{u_1} t_{u_2}) - (1 - t_v) \\ &= t_v - 2t_{u_1} t_{u_2} + 1.\end{aligned}$$

Note that t_v is the product of t_u over all children u of v. Thus, we have $t_v < (t_{u_1} t_{u_2})^{k/2}$, which implies $\Delta C < z^k - 2z^2 + 1$, where $\frac{k-1}{k} \le z \le 1$.

It is easy to verify that $z^6 - 2z^2 + 1 < 0$ for $5/6 \le z \le 1$. Because the value of $z^k - 2z^2 + 1$ decreases with k for fixed z, we see that $\Delta C < 0$ for any $k \ge 6$ and $\frac{k-1}{k} \le z \le 1$, which proves the lemma. □

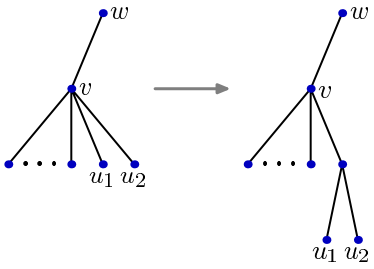

FIG. 2. *Tree transformation* 2.

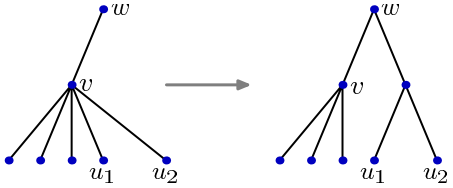

FIG. 3. *Tree transformation* 3.

THEOREM 2. *In a* (p, n)*-optimal tree,*

(1) *any internal node other than the root must have a branching degree of* ≤ 4*;*

(2) *the size of any subtree* T_v*, where* v *is a child of the root, is upper bounded by* $\max\{4(\log q^{-1})^{-1}, 1\}$.

Proof. By using Lemma 5, we can complete the proof of (1) by showing that the optimal tree does not have any internal node with a branching degree of 5.

Assume there is a nonroot internal node v with five children $u_1, \ldots, u_5$. For simplicity, we write t_{u_i} as t_i and assume $t_1 \geq \cdots \geq t_5$. First, observe that $z^5 - 2z^2 + 1 < 0$ when $z \geq 0.86$. According to Lemma 4 and the proof of Lemma 5, it must be the case that both conditions $t_1 t_2 < (0.86)^2 = 0.7396$ and $0.8 \leq t_i < 0.86$ for $2 \leq i \leq 5$ hold. We now prove that under these conditions, another tree transformation will reduce the total cost which contradicts the tree's optimality. We transform the optimal tree into tree T' as shown in Figure 3. By doing so, we increase the total cost by

$$\begin{aligned}\Delta C &= 3(1 - t_3 t_4 t_5) + 2(1 - t_1 t_2) + (1 - t_w) - 5(1 - t_v) \\ &< -2t_1 t_2 - 3t_3 t_4 t_5 + 5t_v + 1 \\ &= (5t_3 t_4 t_5 - 2)t_1 t_2 - (3t_3 t_4 t_5 - 1),\end{aligned}$$

where t_v represents the value before transformation. By using the fact that $t_i \geq 0.8$ for $1 \leq i \leq 5$ and $t_1 t_2 < 0.7396$, it can be verified that $\Delta C < 0$. This completes the proof of property (1).

For property (2), as we have shown in Lemma 4, any child u of v satisfies $q(u) = q^{N_u} > 1/2$. Thus, $N_u < (\log q^{-1})^{-1}$. Since v has a branching degree of at most 4 by property (1) and also $N_v \geq 1$, we have $N_v \leq \max\{4(\log q^{-1})^{-1}, 1\}$. This completes the proof of the theorem. ∎

5. Algorithm for constructing the optimal tree. We will construct a (p, n)-optimal tree by assembling a forest of suitable subtrees. We first generalize the cost function from trees to forests as follows.

DEFINITION 1. *For $L \le n$, we define a (p, n, L)-forest to be a forest of key trees with L leaves in total. The cost of the tree edges in the forest are defined as before, while the cost of the forest is the sum of individual tree costs plus $k \cdot (1 - q^n)$, where k is the number of trees in the forest. We refer to the (p, n, L)-forest with minimum cost as the optimal (p, n, L)-forest and denote that minimum cost by $F(p, n, L)$.*

THEOREM 3. *For any fixed p, Algorithm 1 computes the (p, n)-optimal tree cost in $O(n)$ time.*

Proof. Based on Theorem 2, in a (p, n)-optimal tree, any subtree T_v, where v is a child of the root, satisfies (1) its size is at most $\max\{4(\log q^{-1})^{-1}, 1\}$ and (2) the branching degree of any internal node in T_v is at most 4. For fixed q, we view $4(\log q^{-1})^{-1}$ as a constant and denote it by K. For each i, where $2 \le i \le K$, we consider the minimum cost $R(i)$ of any tree T_v with size i and subject to the degree restriction stated in (2). The value of $R(i)$ can be computed in constant time as follows. First, define (k_1, k_2, k_3, k_4) to be an *i-quadruple* if $k_1 + k_2 + k_3 + k_4 = i$, $0 \le k_1, k_2, k_3, k_4 \le i$, and $k^* \ge 2$, where k^* is the number of positive elements in (k_1, k_2, k_3, k_4). Then $R(i)$ is the minimum value of $R(k_1) + R(k_2) + R(k_3) + R(k_4) + (1 - q^i) \cdot k^*$ over all i-quadruples (k_1, k_2, k_3, k_4), and it can be computed in $O(K^3)$ time using dynamic programming. Now, we can obtain the true (p, n)-optimal tree also by dynamic programming, by computing optimal (p, n, L)-forests as subproblems of size L for $1 \le L \le n$ as given in Algorithm 1. Thus, the total running time of the algorithm is $O(n \cdot K + K^4)$ which is $O(n)$ for fixed p. Algorithm 1 focuses on computing the optimal tree cost; the tree structure can be obtained by keeping track of the optimal branching at every dynamic programming iteration. ∎

Algorithm 1 *Computing optimal key tree*

Input: n and p $(0 \le p \le 1)$
Output: Optimal tree cost $OPT(n, p)$

```
q = 1 − p
K = min{4(log q^{-1})^{-1}, n}
if q ≤ 3^{-1/3} or 2 ≤ n ≤ 4 then
   OPT(p, n) ← n · (1 − q^n)
   Return OPT(p, n)
end if
R(1) = 0
for i = 2 to 4 do
   R(i) = i ∗ (1 − q^i)
end for
i = 5
while i < K do
   Compute R(i), cost of the restricted (p, i)-optimal tree.
   i = i + 1
end while
F(p, n, 0) ← 0
for L = 1 to n do
   F(p, n, L) ← min(R(j) + 1 − q^n + F(p, n, L − j)) over all j, 1 ≤ j ≤ min{K, L}.
end for
OPT(n, p) ← F(p, n, n)
Return OPT(n, p)
```

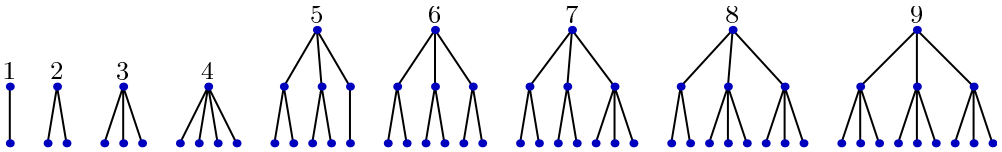

FIG. 4. *Optimal small trees.*

6. Optimal trees as $p \to 0$. Algorithm 1 has running time $O(n \cdot K + K^4)$, where K is upper bounded by $4(\log q^{-1})^{-1}$ (and also by n). We can regard $O(K^4)$ as a constant term for a fixed value of p, but its values get large as $p \to 0$. Therefore, in this final section we will study the structure of (p, n)-optimal trees as $p \to 0$. It turns out the structure of (p, n)-optimal trees depends rather critically on certain number-theoretic properties of n. (Some of the detailed computations will be suppressed.) Suppose $T = T(n)$ denotes a rooted tree with n leaves and edge set E.

For convenience, we let $L(e) = N_v$ if $e = (u, v)$ and u is a child of v. We express the cost of T as

$$P_T(p) = C(T) = \sum_{e \in E} (1 - (1-p)^{L(e)}).$$

Of course, $P_T(0) = 0$ and $P_T(1) = |E|$. The optimal trees as $p \to 0$ are those with $P_T(p)$ having the smallest slope at $p = 0$. Any such optimal tree will remain optimal for an interval $[0, c]$ for some (small) $c > 0$. The slope of $P_T(p)$ at $p = 0$ is denoted by λ_T and can be expressed as

$$\lambda_T = \sum_{e \in E} L(e).$$

We let $\lambda^*(n)$ denote the smallest possible value of λ_T over all trees $T = T(n)$ having n leaves. Our first task will be to determine the exact value of $\lambda^*(n)$ for all values of n.

To begin with, it is easy to check by hand that the trees shown in Figure 4 are optimal for the values $1 \le n \le 9$. This implies the corresponding values of $\lambda^*(n)$ shown below:

n	1	2	3	4	5	6	7	8	9
$\lambda^*(n)$	0	4	9	16	23	30	38	44	54

For integers $t \ge 0$, define the sets $I_t = \{3^t, 3^t + 1, \ldots, 2 \cdot 3^t\}$ and $J_t = \{2 \cdot 3^t, 2 \cdot 3^t + 1, \ldots, 3^{t+1}\}$. Notice that $\lambda^*(n)$ is *linear* on I_0, I_1, J_0, and J_1. We will show that this holds in general for all I_t and J_t.

First, let us extend $\lambda^*(n)$ to a real function $\lambda^*(x)$ for all $x \ge 1$ by linear interpolation (see Figure 5).

The basic recurrence that $\lambda^*(n)$ satisfies is

$$\lambda^*(n) = \min_{2 \le r \le n} \left\{ rn + \sum \lambda^*(i_k) \right\}, \tag{6.1}$$

where the sum is taken over all $i_k \ge 1$ such that $i_1 + \cdots + i_r = n$ and r denotes the degree of the root ρ.

What we will prove is in the following lemma.

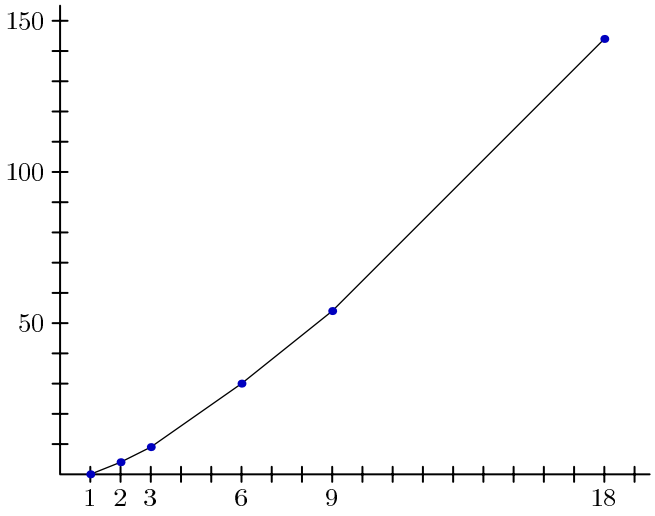

FIG. 5. *Graph of* $\lambda^*(n)$.

LEMMA 6.

$$\lambda^*(x) = \begin{cases} (3t+4)x - 4\cdot 3^t & \text{if} \quad x \in I_t, \\ (3t+5)x - 6\cdot 3^t & \text{if} \quad x \in J_t. \end{cases} \tag{6.2}$$

Proof. It is easy to check that this holds for $t = 0$ and 1. Assume that this holds for some value $t \geq 1$. We also assume that $\lambda^*(x)$ is linear on I_s and J_s, $s \leq t$, and strictly convex between intervals, (i.e., the slopes are strictly increasing on successive intervals). Thus, for each fixed value of r, the sum $\sum \lambda^*(i_k)$ is minimized by taking all the i_k to be as equal as possible. In fact, we can take all $i_k = \frac{n}{r}$, by the assumption of linearity of $\lambda^*(x)$ on the I_s and J_s, $s \leq t$. (Here we need the elementary fact that if $\frac{n}{r} \in [a, a+1]$ for some integer a, then n can be expressed as the sum of u a's and $r-u$ $a+1$'s, for some u with $0 \leq u \leq r$.) Thus, we can write

$$\lambda^*(n) = \min_{2\leq r\leq n} \left\{ rn + r\lambda^* \left(\frac{n}{r}\right) \right\}.$$

Our next job is to eliminate values of r as candidates for achieving the minimum. For example, let us show that

$$5n + 5\lambda^* \left(\frac{n}{5}\right) > 4n + 4\lambda^* \left(\frac{n}{4}\right).$$

Take $n \in I_t$ and consider $\frac{n}{4}$ and $\frac{n}{5}$. Since their ratio is $\frac{5}{4}$, then there are only four possibilities (for some $s < t$):

(i) $\frac{n}{5}, \frac{n}{4} \in \bar{I}_s$;
(ii) $\frac{n}{5}, \frac{n}{4} \in \bar{J}_s$;
(iii) $\frac{n}{5} \in \bar{I}_s$, $\frac{n}{4} \in \bar{J}_s$;
(iv) $\frac{n}{5} \in \bar{J}_s$, $\frac{n}{4} \in \bar{I}_{s+1}$,

where we use $\bar{I}_s$ and $\bar{J}_s$ to denote the intervals $[3^s, 2\cdot 3^s]$ and $[2\cdot 3^s, 3^{s+1}]$, respectively.

In case (i), we have

$$5\lambda^* \left(\frac{n}{5}\right) = 5(3s+4)\frac{n}{5} - 5\cdot 4\cdot 3^s = (3s+4)n - 20\cdot 3^s$$

and

$$4\lambda^*\left(\frac{n}{4}\right) = 4(3s+4)\frac{n}{4} - 4\cdot 4\cdot 3^s = (3s+4)n - 16\cdot 3^s.$$

Thus,

$$5n + 5\lambda^*\left(\frac{n}{5}\right) - 4n - 4\lambda^*\left(\frac{n}{4}\right) = n - 4\cdot 3^s \geq (5-4)3^s = 3^s > 0$$

since $n \geq 5\cdot 3^s$ (because $\frac{n}{5} \in \bar{I}_s$).

Similarly, for case (ii) we have

$$5\lambda^*\left(\frac{n}{5}\right) = (3s+5)n - 30\cdot 3^s$$

and

$$4\lambda^*\left(\frac{n}{4}\right) = (3s+5)n - 24\cdot 3^s.$$

Thus,

$$5n + 5\lambda^*\left(\frac{n}{5}\right) - 4n - 4\lambda^*\left(\frac{n}{4}\right) = n - 6\cdot 3^s \geq (10-6)3^s = 4\cdot 3^s > 0$$

since $n \geq 10\cdot 3^s$ (because $\frac{n}{5} \in \bar{J}_s$).

For case (iii), we have

$$5\lambda^*\left(\frac{n}{5}\right) = (3s+4)n - 20\cdot 3^s$$

and

$$4\lambda^*\left(\frac{n}{4}\right) = (3s+5)n - 24\cdot 3^s.$$

Thus,

$$5n + 5\lambda^*\left(\frac{n}{5}\right) - 4n - 4\lambda^*\left(\frac{n}{4}\right) = 4\cdot 3^s > 0.$$

Finally, for case (iv) we have

$$5\lambda^*\left(\frac{n}{5}\right) = (3s+5)n - 30\cdot 3^s$$

and

$$4\lambda^*\left(\frac{n}{4}\right) = (3s+7)n - 48\cdot 3^s.$$

Thus,

$$5n + 5\lambda^*\left(\frac{n}{5}\right) - 4n - 4\lambda^*\left(\frac{n}{4}\right) = -n + 18\cdot 3^s \geq 3\cdot 3^s > 0$$

since $\frac{n}{5} \in \bar{J}_s \Longrightarrow n \leq 15\cdot 3^s$.

Hence, in all cases it is better to use $r = 4$ than $r = 5$; i.e., the value of r which determines $\lambda^*(n)$ for $n \in I_t$ cannot be 5. A similar argument rules out *any* value of $r \geq 5$ (we omit the calculations which are very similar to the case we have just

done). Also, it is easy to see that the same arguments apply if we initially assumed that $n \in J_{t+1}$, where now we can assume that the induction hypothesis holds for all $s \le t$, and for I_{t+1}, as well.

Thus, we are left with the possibilities that $r = 2, 3$, or 4. Here, things become a little more interesting! When we apply the preceding argument to compare $3n + \lambda^*(n/3)$ and $4n + \lambda^*(n/4)$, we find that the difference is positive in cases (ii) and (iii), but can be 0 in cases (i) and (iv) exactly when $n = 4 \cdot 3^{t-1}$.

For $r = 2$, the same arguments show that there is a *whole interval* of values for n where the difference $2n + 2\lambda^*(\frac{n}{2}) - 3n - 3\lambda^*(\frac{n}{3})$ can be 0, namely when $4 \cdot 3^t \le n \le 6 \cdot 3^t$ (it can never be negative).

With this information, we can now compute the values of $\lambda^*(n)$ for $n \in I_{t+1} \cup J_{t+1}$. When we do this and extend to the real function $\lambda^*(x)$, we find that (6.2) holds for $t+1$. With this, the induction step is completed, and we have shown that (6.2) holds for all t. □

In particular, we have

$$\lambda^*(3^t) = t \cdot 3^{t+1}, \quad \lambda^*(2 \cdot 3^t) = (6t+4) \cdot 3^t, \quad \lambda^*(4 \cdot 3^t) = (12t+16) \cdot 3^t. \tag{6.3}$$

Let $T^*(n)$ denote an optimal tree with n leaves as $p \to 0$. Although we now know the *slope* $\lambda^*(n)$ of $P_{T^*(n)}$, we don't yet know the degree of the root of $P_{T^*(n)}$ in the case that n is in the "ambiguous" range $4 \cdot 3^t \le n \le 6 \cdot 3^t$. This will depend on the second derivative of $P_T(p)$ evaluated at $p = 0$. This is just

$$-\sum_{e \in E} \binom{L(e)}{2}.$$

Since this is negative, we want to make the sum $\sum_{e \in E} \binom{L(e)}{2}$ as large as possible in order to find the optimal tree $T^*(n)$. Let $\mu^*(n)$ denote the largest possible value of this sum over all trees $T(n)$ for which $\lambda_{T(n)} = \lambda^*(n)$. We know, in general, that an optimal tree $T^*(n)$ with root degree r has on each of its root edges an optimal subtree $T^*(i_k)$, where

$$\sum_{k=1}^{r} i_k = n$$

and *all the i_k must lie in the same interval I_t (or J_t)*, since otherwise the optimal value $\lambda^*(n)$ would not be achieved.

First, observe that we know the optimal tree $T^*(3^t)$ since it must have a root degree of 3, so by induction we can deduce that $\mu^*(3^t) = \frac{1}{4}(3^{2t+2} - (2t+3)3^{t+1})$. Now, for $T^*(2 \cdot 3^t)$, either the root has degree 2, in which case the two subtrees must both be $T^*(3^t)$'s, or the root has degree 3, in which case the three subtrees are all $T^*(2 \cdot 3^{t-1})$'s (since in both cases, the subtree sizes are endpoints of an I or J interval). By induction, we can conclude that degree 3 wins here and that $\mu^*(2 \cdot 3^t) = 3^{2t+2} - (3t+7)3^t$. Finally, for $T^*(4 \cdot 3^t)$ (where there are three possible choices for the root degree), we find that degree 4 wins in this case, and we have $\mu^*(4 \cdot 3^t) = 41 \cdot 3^{2t} - (6t+17)3^t$. These are the only values of n for which a root degree of 4 is optimal.

The remaining problem is to eliminate the possibility of a root degree of 2 for the values $4 \cdot 3^t < n \le 6 \cdot 3^t$. It should be noted that to obtain the largest possible $\mu^*(n)$, the subtree sizes will tend to be as far apart as possible (consistent with staying in

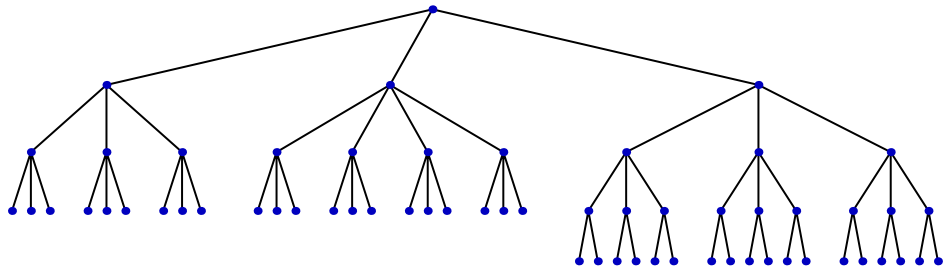

FIG. 6. *An optimal* $T^*(39)$.

the same I or J interval), again because of the tendency for $\mu^*(n)$ to be convex. It isn't convex everywhere (or even monotone), however, because of the unusually large values at $4 \cdot 3^t$. After all, since

$$\mu^*(n) = \max_{2 \le r \le 4} \left(r\binom{n}{2} + \sum_{i_1 \ldots i_r} \mu^*(i_k) \right),$$

then when $r = 4$, we get an especially large contribution from the term $r\binom{n}{2}$.

First, let us deal with $n \in J_t$, i.e., $2 \cdot 3^t \le n \le 3^{t+1}$. In this case, $T^*(n)$ has a root degree of 3 and so all three (optimal) subtrees have sizes $i_k \in J_{t-1}$. We put the proof of the following lemma in the appendix.

LEMMA 7. *Let* $n = 2 \cdot 3^t + r$ *with* $0 \le r \le 3^t$. $\max\{\mu^*(i_1) + \mu^*(i_2) + \mu^*(i_3) : i_1 + i_2 + i_3 = n,$ *all* $i_k \in J_{t-1}\}$ *occurs when the* i_k *are "maximally spread," i.e., when at least two of the* i_k *are equal to the endpoint values* $2 \cdot 3^{t-1}$ *and* 3^t *of* J_{t-1}.

We derive in the following lemma that the optimal tree as $p \to 0$ cannot have a root degree of 2. Its proof is also put in the appendix.

LEMMA 8. $T^*(n)$ *cannot have a root degree of* 2 *for any* $n > 2$.

We summarize what we have shown in the following result.

THEOREM 4. *As* $p \to 0$, *the* (p, n)*-optimal tree* $T^*(n)$ *always has a root degree of* 3 *except for* n *of the form* $4 \cdot 3^t$, *in which case* $T^*(4 \cdot 3^t)$ *has a root degree of* 4, *and for* $n = 2$, *when* $T^*(2)$ *has a root degree of* 2. *When* $2 \cdot 3^t \le n \le 3^{t+1}$, *then* $T^*(n)$ *is as close to a balanced ternary tree with* n *leaves as possible. Namely, all subtrees (as well as* $T^*(n)$ *itself) have root degrees of* 3, *except for the very bottom level, where subtrees of size* 2 *can occur. However, when* $3^t \le n < 2 \cdot 3^t$, $T^*(n)$ *can deviate substantially from a balanced ternary tree.*

As an example for the situation $3^t \le n \le 2 \cdot 3^t$, note that $T^*(39)$ has three subtrees of sizes $9, 12$, and 18 (see Figure 6). In general, it seems to be difficult to predict the sizes of the subtrees in the optimal tree $T^*(n)$ for certain values of n. For example, for $n = 1252$, the sizes are $280, 486$, and 486, while for $n = 1253$, the sizes are $324, 443$, and 486. This is an example of the effect of a number of the form $324 = 4 \cdot 3^4$ having an especially large value of μ^*, thereby causing a preferential bias towards using it. In this case, $\mu^*(324) = 265680$ while $\mu^*(323) = 240997, \mu^*(325) = 244014$. In fact, we already see this happening at $n = 11$, where the optimal tree $T^*(11)$ has subtree sizes $3, 4$, and 4, which are not maximally spread in I_1.

Appendix.

Proof of Lemma 7. This is true by inspection for $t = 0, 1$. Suppose it holds for some value of $t \geq 1$ and let $n = 2 \cdot 3^{t+1} + r \in J_{t+1}$. Now, by (1),

$$\mu^*(n) = 3\binom{n}{2} + \max\{\mu^*(i_1) + \mu^*(i_2) + \mu^*(i_3) : i_1 + i_2 + i_3 = n, \text{ all } i_k \in J_t\}. \tag{A.1}$$

By induction, the maximum is achieved when the i_k are maximally spread. In particular, this means that, assuming $i_1 \leq i_2 \leq i_3$:

(a) if $0 \leq r \leq 3^t$, then $i_1 = 2 \cdot 3^t$, $i_2 = 2 \cdot 3^t$, $i_3 = 2 \cdot 3^t + r'$;
(b) if $3^t \leq r \leq 2 \cdot 3^t$, then $i_1 = 2 \cdot 3^t$, $i_2 = 2 \cdot 3^t + r'$, $i_3 = 3^{t+1}$;
(c) if $2 \cdot 3^t \leq r \leq 3^{t+1}$, then $i_1 = 2 \cdot 3^t + r'$, $i_2 = 3^{t+1}$, $i_3 = 3^{t+1}$.

Let $\Delta(m)$ denote the difference $\mu^*(m+1) - \mu^*(m)$. Then this implies, for $0 \leq r < 3^{t+1}$, the fundamental equation

$$\Delta(2 \cdot 3^{t+1} + r) = \Delta(2 \cdot 3^t + r') + 3n, \tag{A.2}$$

where $0 \leq r' < 3^t$ and $r \equiv r' \pmod{3^t}$. The term $3n$ comes from the difference $3\binom{n+1}{2} - 3\binom{n}{2} = 3n$. From this it follows (by induction) that a sum of k consecutive values

$$\Delta(2 \cdot 3^{t+1} + u) + \Delta(2 \cdot 3^{t+1} + (u+1)) + \cdots + \Delta(2 \cdot 3^{t+1} + (u+k-1))$$

for $0 \leq u \leq 3^{t+1} - k + 1$ is *minimized* by taking $u = 0$ since the sum is a monotone function of u. From this, the claim now follows, since any choice of the i_k which isn't maximally spread can be replaced by a maximally spread choice which can only increase the value of $\mu^*(i_1) + \mu^*(i_2) + \mu^*(i_3)$ (the difference in the two values being equal to the difference of two interval sums of the Δ's).

The next step is to obtain an explicit expression for the value $\Delta(n)$ for $n = 2 \cdot 3^t + r$ with $0 \leq r < 3^t$. We do this by iterating (5). The result is

$$\Delta(2 \cdot 3^t + r) = 3^{t+2} - 2 + 3R_t(r), \tag{A.3}$$

where $R_t(r)$ is defined as follows. Write r in its base 3 expansion as $r = r_{t-1}r_{t-2}r_{t-3}\ldots r_2 r_1 r_0$, where each $r_k \in \{0, 1, 2\}$. Then $R_t(r)$ is the sum of the t numbers corresponding to the t rows (read in base 3) of the array

$$\begin{array}{ccccccc}
r_{t-1} & r_{t-2} & r_{t-3} & \ldots & r_2 & r_1 & r_0 \\
0 & r_{t-2} & r_{t-3} & \ldots & r_2 & r_1 & r_0 \\
0 & 0 & r_{t-3} & \ldots & r_2 & r_1 & r_0 \\
0 & 0 & 0 & \ldots & r_2 & r_1 & r_0 \\
 & & & \vdots & & & \\
0 & 0 & 0 & \ldots & r_2 & r_1 & r_0 \\
0 & 0 & 0 & \ldots & 0 & r_1 & r_0 \\
0 & 0 & 0 & \ldots & 0 & 0 & r_0.
\end{array}$$

Thus, $R_t(0) = 0$, $R_t(1) = 3t$, $R_t(14) = 14t - 21$, etc.

From this we can write down an "explicit" expression for $\mu^*(n)$ for $n = 2 \cdot 3^t + r \in J_t$, which is

$$\begin{aligned}
\mu^*(2 \cdot 3^t + r) &= \mu^*(2 \cdot 3^t) + \Delta(2 \cdot 3^t) + \Delta(2 \cdot 3^t + 1) + \cdots + \Delta(2 \cdot 3^t + r - 1) \\
&= 3^{2t+2} - (3t+7)3^t + r \cdot (3^{t+2} - 2) + 3\sum_{k=0}^{r-1} R_t(k). \qquad \square
\end{aligned} \tag{A.4}$$

Proof of Lemma 8. As we have seen, this is only possible when $4 \cdot 3^t \le n \le 6 \cdot 3^t$. For such n, $2 \cdot 3^t \le \frac{n}{2} \le 3^{t+1}$. This implies that if $T^*(n)$ has a root degree of 2, then the two subtrees $T^*(i_1)$ and $T^*(i_2)$ with $i_1 + i_2 = n$ must have i_1 and i_2 both in J_t and be maximally spread (by the same inductive argument as before), which in this case means that at least one of the i_k must be one of the endpoint values $2 \cdot 3^t$ or 3^{t+1} of J_t. More precisely, if $T^{(2)}(n)$ denotes the best tree on n leaves with a root degree of 2 (and, of course, achieving the optimal value of $\lambda^*(n)$), and we assume that $i_1 \le i_2$, then

(a) if $4 \cdot 3^t \le n = 4 \cdot 3^t + r_1 \le 5 \cdot 3^t$, then $i_1 = 2 \cdot 3^t$, $i_2 = 2 \cdot 3^t + r_1$;
(b) if $5 \cdot 3^t \le n = 5 \cdot 3^t + r_2 \le 6 \cdot 3^t$, then $i_1 = 2 \cdot 3^t + r_2$, $i_2 = 3^{t+1}$.

In either case, we have

$$\mu_{T^{(2)}(n)} = 2\binom{n}{2} + \mu^*(i_1) + \mu^*(i_2), \tag{A.5}$$

where we know exactly the values of $\mu^*(i_1)$ and $\mu^*(i_2)$, since both arguments are in J_t. What we would like to do is compare this with the best tree $T'(n)$ with a root degree of 3 and show that $\mu_{T'(n)} > \mu_{T^{(2)}(n)}$. Unfortunately, we don't know the *best* tree $T'(n)$ with a root degree of 3. However, we know a pretty good one. This is the tree for which all of its subtrees also have root degrees of 3, recursively, until down to the very last level, in which any subtree with only 2 leaves must have a root degree of 2. For this family of trees $T'(n)$, we let $\mu'(n)$ denote the corresponding value of the second derivative (with its sign changed), and we let $\Delta'(m) = \mu'(m+1) - \mu'(m)$. Thus, $\Delta(n)$ and $\Delta'(n)$ agree for $n = 2, 3, 4, 5, 7, 8, 9$ while we have $\Delta(6) = 24$ and $\Delta'(6) = 20$.

More important is that $\Delta'(n)$ satisfies the same recurrence (5) that $\Delta(n)$ does, namely,

$$\Delta'(3^{t+1} + r) = \Delta'(3^t + r') + 3n, \tag{A.6}$$

where $n = 3^{t+1} + r$, $0 \le r < 3^{t+1}, 0 \le r' < 3^t$, and $r \equiv r' \pmod{3^t}$. This follows from the same arguments which established (5). Iterating this, we obtain the analogue of (6):

$$\Delta'(3^t + r) = \frac{3^{t+2} - 5}{2} + R_t(r). \tag{A.7}$$

Finally, summing this over r and using the fact that $\mu(3^t) = \frac{1}{4}(3^{2t+2} - (6t+9)3^t)$, we obtain the following analogue of (7):

$$\mu'(3^t + r) = \frac{1}{4}\big(3^{2t+2} - (6t+9)3^t\big) + \frac{r(3^{t+2} - 5)}{2} + 3\sum_{k=0}^{r-1} R_t(k). \tag{A.8}$$

We are now in a position to complete the argument. Let $n = 4 \cdot 3^t + r$. There are two cases.

Case 1. $0 \le r \le 3^t$. In this case

$$\mu'(n) = \mu^*(3^t) + \mu^*(2 \cdot 3^t) + \mu'(3^t + r) + 3\binom{n}{2}$$

and

$$\mu_{T^{(2)}(n)} = \mu^*(2 \cdot 3^t) + \mu^*(2 \cdot 3^t + r) + 2\binom{n}{2}.$$

Hence,

$$\mu'(n) > \mu_{T^{(2)}(n)}$$

if and only if

$$\mu^*(3^t) + \mu^*(2\cdot 3^t) + \mu'(3^t + r) + 3\binom{n}{2} > \mu^*(2\cdot 3^t) + \mu^*(2\cdot 3^t + r) + 2\binom{n}{2}$$

if and only if

$$\frac{3^{2t+2} - (6t+9)3^t}{2} + \frac{r(3^{t+2} - 5)}{2} + 3\sum_{k=0}^{r-1} R_t(k) + \binom{n}{2}$$

$$> 3^{2t+2} - (3t+7)3^t + r(3^{t+2} - 2) + 3\sum_{k=0}^{r-1} R_t(k)$$

if and only if

$$2\binom{4\cdot 3^t + r}{2} > 3^{2t+2} + r3^{t+2} - 5\cdot 3^t + r,$$

which is easily verified. This finishes Case 1.

Case 2. $3^t \le r \le 2\cdot 3^t$. Let $s = r - 3^t$. In this case

$$\mu'(n) = \mu'(3^t + s) + \mu^*(2\cdot 3^t) + \mu^*(2\cdot 3^t) + 3\binom{n}{2}$$

and

$$\mu_{T^{(2)}(n)} = \mu^*(2\cdot 3^t + s) + \mu^*(3^{t+1}) + 2\binom{n}{2}.$$

Hence,

$$\mu'(n) > \mu_{T^{(2)}(n)}$$

if and only if

$$\mu'(3^t + s) + \mu^*(2\cdot 3^t) + \mu^*(2\cdot 3^t) + \binom{n}{2} > \mu^*(2\cdot 3^t + s) + \mu^*(3^{t+1})$$

if and only if

$$\frac{3^{2t+2} - (6t+9)3^t}{4} + \frac{s(3^{t+2} - 5)}{2} + 3\sum_{k=0}^{r-1} R_t(k) + 2(3^{2t+2} - (3t+7)3^t) + \binom{5\cdot 3^t + s}{2}$$

$$> \ 3^{2t+2} - (3t+7)3^t + s(3^{t+2} - 2) + 3\sum_{k=0}^{r-1} R_t(k) + \frac{3^{2t+4} - (6t+15)3^{t+1}}{4}$$

if and only if

$$2\binom{5\cdot 3^t + s}{2} > 18\cdot 3^{2t} - 4\cdot 3^t + s(3^{t+2} + 1).$$

As before, it is easy to check that this inequality holds, and Case 2 is completed.

Thus, we have shown that in the ambiguous range $4 \cdot 3^t \leq n < 6 \cdot 3^t$, there is always a (ternary) tree $T'(n)$ which dominates the best tree $T^{(2)}(n)$ with a degree 2 root (where all trees under consideration must achieve the optimal value of $\lambda^*(n)$). Consequently, the *optimal* tree $T^*(n)$ also dominates this $T^{(2)}(n)$ as well. □

REFERENCES

[1] M. T. Goodrich, J. Z. Sun, and R. Tamassia, *Efficient tree-based revocation in groups of low-state devices*, in Advances in Cryptology—CRYPTO, Lecture Notes in Comput. Sci. 3152, Springer, Berlin, 2004, pp. 511–527.

[2] X. S. Li, Y. R. Yang, M. G. Gouda, and S. S. Lam, *Batch re-keying for secure group communications*, in Proceedings of the Tenth International Conference on the World Wide Web, WWW10, Hong Kong, 2001, pp. 525–534.

[3] D. M. Wallner, E. G. Harder, and R. C. Agee, *Key management for multicast: Issues and architectures*, in internet draft *draft-waller-key-arch-*01.*txt*, Sept. 1998

[4] C. K. Wong, M. G. Gouda, and S. S. Lam, *Secure group communications using key graphs*, IEEE/ACM Trans. Networking, 8 (2003), pp. 16–30.

[5] F. Zhu, A. Chan, and G. Noubir, *Optimal tree structure for key management of simultaneous join/leave in secure multicast*, in Proceedings of the IEEE Military Communication Conference (MILCOM), Boston, 2003, pp. 773–778.