Fast Algorithms for Constructing Maximum Entropy Summary Trees*

Richard Cole[1,**] and Howard Karloff[2]

[1] Computer Science Department, Courant Institute, NYU
[2] Yahoo! Research

Abstract. Karloff and Shirley recently proposed "summary trees" as a new way to visualize large rooted trees (Eurovis 2013) and gave algorithms for generating a maximum-entropy k-node summary tree of an input n-node rooted tree. However, the algorithm generating optimal summary trees was only pseudo-polynomial (and worked only for integral weights); the authors left open existence of a polynomial-time algorithm. In addition, the authors provided an additive approximation algorithm and a greedy heuristic, both working on real weights.

This paper shows how to construct maximum entropy k-node summary trees in time $O(k^2n + n\log n)$ for *real* weights (indeed, as small as the time bound for the greedy heuristic given previously); how to speed up the approximation algorithm so that it runs in time $O(n + (k^4/\epsilon)\log(k/\epsilon))$, and how to speed up the greedy algorithm so as to run in time $O(kn + n\log n)$. Altogether, these results make summary trees a much more practical tool than before.

1 Introduction

How should one draw a large n-node rooted tree on a small sheet of paper or computer screen? Recently, in Eurovis 2013, Karloff and Shirley [4] proposed a new way to visualize large trees. While the best introduction to summary trees appears in [4], here we give a necessarily short description. A user has an n-node node-weighted tree T and wants to draw a k-node summary S of T on a small screen or sheet of paper, k being user-specified. We begin with an informal, bottom-up, operational description. Two type of contraction are performed: subtrees are contracted to single nodes that represent the corresponding subtrees; similarly multiple sibling subtrees (subtrees whose roots are siblings) are contracted to single nodes representing them. The node resulting from the latter contraction is called a group node. The one constraint is that each node in the summary tree have at most one child that is a group node. An example is shown in Figure 1 (based on a figure in [4]).

Next, we give a more formal description. Let T_v denote the subtree of T rooted at v. We name each node of S by the set of nodes of T that it represents.

* A full version of the paper is available at `http://arxiv.org/abs/1404.5660`

** Richard Cole's work was supported in part by NSF grant CCF-1217989.

J. Esparza et al. (Eds.): ICALP 2014, Part I, LNCS 8572, pp. 332–343, 2014.

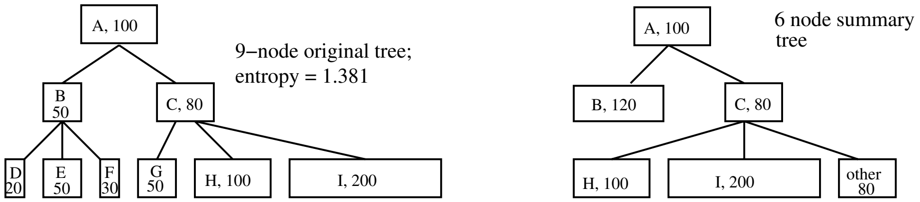

Fig. 1. On the left, a 9-node tree with node weights, and to the right, a 6-node summary tree of the original 9-node tree

The following comprise the possible *summary trees for* T_v: If T_v has just one node, the only summary tree is the one node $\{v\}$. Otherwise, a summary tree for T_v is one of:

1. a one-node tree $V(T_v)$ (the set of nodes in T_v); or
2. a singleton node $\{v\}$ and summary trees for the subtrees rooted at the children of v (and edges from $\{v\}$ to the roots of these summary trees); or
3. a singleton node $\{v\}$, a node $other_v$ representing a non-empty subset U_v of v's children and all the descendants of the nodes $x \in U_v$, and for each of v's children $x \notin U_v$ a summary tree for T_x (and edges from $\{v\}$ to $other_v$ and to the roots of the summary trees for each T_x).[1] Sometimes we will overload the term $other_v$ by using it to denote the subset U_v.

We allow arbitrary nonnegative real weights w_v on the nodes v of the input tree T. The weight of a node in a summary tree is defined to be the sum of the weights of the corresponding nodes in T. Paper [4] defined the entropy of a k-node summary tree with nodes of weights $W_1, W_2, ..., W_k$ to be $-\sum_{i=1}^{k} p_i \lg p_i$, where $p_i = W_i/W$ and W is the sum of all node weights, the usual information-theoretic entropy. Paper [4] then proposed that the most informative summary trees are those of maximum entropy. As noted in [4], this is a natural way to think about the information contained in a node-weighted tree. For given a bound on the number of nodes available in a summary tree, it seems plausible that a best summary tree is one of maximum entropy, because it is theoretically the most informative. This provided a principled way to identify the best k-node summary tree, in contrast to more heuristic and operational rules in prior work.

The fact that $other_v$ is an *arbitrary* non-empty subset of v's potentially large set of children is what makes finding maximum entropy summary trees difficult. Indeed, [4] resorted to using a dynamic program over the node weights (which worked provided that the weights were integral) and which led to a final running time of $O(K^2 nW)$, where W is the sum of the node weights and K is the maximum k for which one is interested in finding a k-node summary tree. Given K, the dynamic program finds maximum entropy k-node summary trees for $k = 1, 2, \ldots, K$; from now on we assume that the user specifies K and k-node

[1] $other_v$ sets of size 1 are covered by Cases 2 and 3, but this redundancy is convenient for the algorithm description.

Table 1. Running times of the algorithms; $W_0 = O((K/\epsilon)\log(K/\epsilon))$

	Optimal Entropy	Greedy	ϵ-Approximate
Known results	$O(K^2nW)$ [4]	$O(K^2n + n\log n)$ [4]	$O(K^2nW_0)$ [4]
New results	$O(K^2n + n\log n)$	$O(Kn + n\log n)$	$O(n + K^3W_0 + W_0\log W_0)$

summary trees are found for all $k \le K$. The algorithm worked well when W was small, but failed to terminate on two of the five data sets used in [4].

The key to obtaining a running time independent of W is to develop a fuller understanding of the structure of maximum entropy summary trees. Our new understanding readily yields a truly polynomial-time algorithm. The main remaining challenge is to create and analyze an effective implementation. We give an algorithm running in time $O(K^2n+n\log n)$ [2]; it generates maximum entropy summary trees even for real weights, assuming, of course, a real-arithmetic model of computation, which is necessary (even for integral weights) because of the computation of logarithms. This result is based on a structural theorem which shows that the *other* sets, while allowed to be arbitrary, can be assumed, without loss of generality, to have a simple structure.

To deal with the case of real weights or exceedingly large integral weights, [4] gave an algorithm based on scaling, rounding, and algorithmic discrepancy theory which builds a summary tree whose entropy is within ϵ *additively* of the maximum, in time $O(K^2nW_0)$, where W_0 is $O((K/\epsilon)\log(K/\epsilon))$. Keep in mind here that K is meant to be small, e.g., 100 or 500, while n is meant to go to infinity, and also that W_0 is a function only of K and ϵ (and neither of n nor W). The key here was to show that scaling the real input weights to have sum W_0, rounding them using algorithmic discrepancy theory, and then running the exact dynamic program previously mentioned on the rounded weights caused a loss of only ϵ in the final entropy.

This paper shows that the same algorithm can be implemented in time $O(n+K^3W_0 + W_0\log W_0)$; this is linear time if n is larger than the other terms. The key here is to notice that if the sum of integral weights is W_0, which is small, and $n \gg W_0$, then most nodes have rounded weight 0. Surely one shouldn't have to devote a lot of time to nodes of weight 0, and our algorithm, by effectively replacing n by $O(W_0)$, exploits this intuition.

Last, [4] proposed a fast greedy algorithm to generate summary trees. Running in time $O(K^2n + n\log n)$ (though [4] overlooked the $n\log n$ time needed for sorting), the algorithm never took longer than six seconds to run on the data sets of [4]. This paper shows that a simple modification to the greedy code, neither suggested in [4] nor implemented in the associated C code, specifically, not computing a k-node summary tree of a tree rooted at a node having fewer than k descendants, decreases the running time bound of the greedy algorithm

[2] Actually, this can be reduced to $O(K^2n)$ time by using a combination of fast selection and sorting instead of sorting alone in various places.

from $O(K^2n + n \log n)$ to $O(Kn + n \log n)$. While the modification is trivial, its analysis is not.

Taken together, these new results show that maximum entropy summary trees are a much more practical tool than was previously known.

A number of the proofs are omitted for lack of space. They can be found in the full version of the paper (see http://arxiv.org/abs/1404.5660).

Previous Work. Traditionally tree visualization involved either visualizing the entire tree or allowing the user to interactively specify tree parts of interest. Approaches taken include "Degree-of-interest trees" [2,3], "hyperbolic browsers" [5], and the "accordion drawing technique" [1,7]. "Space-filling" layouts, e.g., treemaps [9], are another popular method. Paper [6] is a recent survey on techniques for drawing large graphs. Also see [4] for other relevant previous work.

2 Structural Theorem

This section proves a structural theorem which implies that maximum entropy summary trees can be computed in polynomial time, in a real-arithmetic model of computation. We begin by relating our approach to the greedy algorithm from [4]. Let v be a node of an input tree and suppose that $\{v\}$ appears in the summary tree. Recall that $other_v$ denotes the group child of v, if any.

Definition 1. *1. The* size s_v *of a node* v *in* T *is the sum of the weights of its descendants.*

2. n_v *denotes the number of descendants of* v *(including* v*).*

3. d_v *denotes the degree of* v*, the number of children it has.*

4. $\langle v_1, v_2, ..., v_{d_v} \rangle$ *denotes the children of* v *when sorted into nondecreasing order by size. (Fix one sorted order for each* v*, breaking ties arbitrarily.)*

5. The prefixes *of* $\langle v_1, v_2, ..., v_{d_v} \rangle$ *are the sequences* $\langle v_1, v_2, ..., v_i \rangle$ *and sets* $\{v_1, v_2, ..., v_i\}$ *for* $i \geq 0$.

The greedy algorithm in [4] sorted and then processed the children of each node in nondecreasing order by size; more about this later. It finds a maximum entropy summary tree among those in which for each v, either $other_v$ does not exist or is a nonempty prefix of $\langle v_1, v_2, ..., v_{d_v} \rangle$, but this need not be the optimal summary tree. In fact, [4] gives a 7-node tree T for which the uniquely optimal 4-node summary tree has an $other_v$ node which is not a prefix of v's children. In their example, the greedy algorithm achieves approximately 1 bit of entropy, but the optimal summary tree achieves approximately 1.5 bits. This example proves that restricting $other_v$ to be a prefix of the list of v's children can lead to summary trees of suboptimal entropy. Consequently, [4] resorted to a pseudo-polynomial-time dynamic program in order to find the optimal *other* sets.

The definition of summary trees allows $other_v$ to represent an *arbitrary* nonempty subset of v's children (and all their descendants). However, in this paper we prove the surprising fact that, without loss of generality, in every summary tree *of maximum entropy*, $other_v$ can be assumed to have a special form, a simple extension of the "prefix" form used in the greedy algorithm from [4].

Definition 2. *The* near-prefixes *of* $\langle v_1, v_2, \ldots, v_{d_v}\rangle$ *are the sequences* $\langle v_1, v_2, \ldots, v_i; v_j\rangle$ *and the sets* $\{v_1, v_2, \ldots, v_i; v_j\}$ *where* $i \geq 0$*,* $j \geq i+2$*, and* $j \leq d_v$*.* v_j *is called the non-prefix element. This terminology is also applied to the sequence* $\langle T_{v_1}, T_{v_2}, \ldots, T_{v_{d_v}}\rangle$ *of trees rooted at* $v_1, v_2, \ldots, v_{d_v}$*, respectively.*

We prove the following structural theorem:

Theorem 1. *For each* k*,* $1 \leq k \leq n$*, there is a maximum entropy* k*-node summary tree* S *in which, for every node* v*,* $other_v$*, when present, is either a prefix or a near-prefix of* $\langle T_{v_1}, T_{v_2}, \ldots, T_{v_{d_v}}\rangle$*.*

Proof. For any summary tree R of an n-node tree T, let $M = 2n+1$ and define $\Phi(R) = \sum_{v:other_v \text{ exists}} M^{n-d_R(v)} \sum_{j:v_j \in other_v} j$, where $d_R(v)$ denotes the depth in R of the node $other_v$. Among all maximum entropy summary trees for T, let S be one for which $\Phi(S)$ is minimum. (The role of Φ will be to enable tie-breaking among equal-weight summary trees.)

Lemma 1. *Let* v *be a node of* T *such that* $other_v$ *exists in* S*. If* $v_i \notin other_v$ *and* $v_j \in other_v$*, where* $i < j$*, then* T_{v_i} *is represented by two or more nodes in* S*.*

Proof. Suppose, for a contradiction, that T_{v_i} is represented by a single node. Consider the following alternate summary tree S': S' is obtained from S by replacing v_j in $other_v$ by v_i, and by representing T_{v_j} by a single node. The number of nodes in the summary tree remains k.

Let s_0 denote the sum of the sizes of all the children of v in $other_v - \{v_j\}$. (Here "$other_v$" refers to $other_v$ before the change.) Then W times the increase in entropy in going from S to S' is given by

$$I = (s_0 + s_{v_i}) \lg \frac{W}{s_0 + s_{v_i}} + s_{v_j} \lg \frac{W}{s_{v_j}} - (s_0 + s_{v_j}) \lg \frac{W}{s_0 + s_{v_j}} - s_{v_i} \lg \frac{W}{s_{v_i}}.$$

The derivative of this term with respect to s_{v_i} is $\lg \frac{s_{v_i}}{s_0 + s_{v_i}} \leq 0$. As $i < j$, $s_{v_i} \leq s_{v_j}$, and thus I is necessarily nonnegative (for it declines to 0 at $s_{v_i} = s_{v_j}$); consequently, there is a nonnegative increase in entropy, and hence S' is also a maximum entropy summary tree. Furthermore, if d is the depth of $other_v$ in S, then $\Phi(S') - \Phi(S) \leq -(j-i)M^{n-d} < 0$, which contradicts the assumption that S is a maximum entropy summary tree of minimum $\Phi(S)$. ■

Lemma 2. *Let* v *be a node in* T *such that* $other_v$ *exists in* S*. If* $v_i \notin other_v$ *and* $v_{i+1} \in other_v$*, then* $v_j \notin other_v$ *for all* $j > i+1$*.*

Proof. Suppose, for a contradiction, that $v_j \in other_v$, for some $j > i+1$.

By Lemma 1, T_{v_i} is represented by two or more nodes in S. Hence $\{v_i\}$ appears as a node in the summary tree, and $\{v_i\}$ has one or more children in S. In S, let x be a descendant of $\{v_i\}$ of maximum depth in S. Node x is a proper descendant of $\{v_i\}$.

We will show now that combining node x with another node in a specified way yields a summary tree of T_{v_i} with one fewer node and having entropy at most s_{v_i} smaller. Node x is not $\{v_i\}$. Let y be x's parent in S. Node $y = \{u\}$ for some node u in T (since every nonleaf in a summary tree represents a single node of T). There are four cases to analyze, but before turning to them, we state the following simple lemma which we will need; it can be proven by calculus.

Lemma 3. *If* $a, b \geq 0$*,* $-a \lg a - b \lg b + (a+b) \lg(a+b) \leq a + b$*.*

Let s_x, for a node x in summary tree S, denote the sum of the weights of all the nodes of T represented by x. (For a node of the form *other*$_v$, we mean the sum of the *sizes* of all the children of v in *other*$_v$, or equivalently, the sum of the weights of all their descendants.)

Now we begin the case analysis. Let d be the depth in S of node $\{v_i\}$.

1. y's only child in S is x.
   We combine nodes x and $y = \{u\}$ into a node z representing T_u. Recall that w_u denotes u's weight. Then W times the entropy decrease equals

$$\begin{aligned} & s_x \lg(W/s_x) + w_u \lg(W/w_u) - (s_x + w_u) \lg(W/(s_x + w_u)) \\ & \quad = \quad - s_x \lg s_x - w_u \lg w_u + (s_x + w_u) \lg(s_x + w_u) \\ & \quad \leq \quad s_x + w_u \quad \text{(by Lemma 3)} \quad = \; s_z \; \leq \; s_{v_i}. \end{aligned}$$

   This change leaves Φ unchanged.
2. x has a sibling in S and *other*$_u$ does not exist.
   Hence x is either $\{\alpha\}$ or T_α for some node $\alpha \in T$.
   We create a new *other*$_u$ node by combining x with an arbitrary sibling x' of x. Because x is of maximum depth in S, x' is either of the form $\{\beta\}$ (node β in T has no children) or T_β, for some β in T. The resulting entropy decrease equals

$$\begin{aligned} & s_x \lg(W/s_x) + s_{x'} \lg(W/s_{x'}) - (s_x + s_{x'}) \lg(W/(s_x + s_{x'})) \\ & \quad = \quad - s_x \lg s_x - s_{x'} \lg s_{x'} + (s_x + s_{x'}) \lg(s_x + s_{x'}) \\ & \quad \leq \quad s_x + s_{x'} \quad \text{(by Lemma 3)} \quad \leq \; s_{v_i}. \end{aligned}$$

   This change can increase Φ by at most $2n \cdot M^{n-(d+1)}$, because the depth of the new *other*$_u$ node is at least $d+1$.
3. x has a sibling in S and $\{x\} =$ *other*$_u$.
   We choose an arbitrary sibling x' of x and add it to *other*$_u$. The entropy calculation is the same as for Case 2. This change can increase Φ by at most $n \cdot M^{n-(d+1)}$, where d is the depth of $\{v_i\}$ in S.
4. x has a sibling in S, *other*$_u$ exists, and and $\{x\} \neq$ *other*$_u$.
   We add x to *other*$_u$. Let x' be the node *other*$_u$. The calculations are exactly the same as in Case 3.

In all four cases, the decrease in entropy is at most s_{v_i} and the increase in Φ is at most $2nM^{n-d-1}$.

Now we show how to generate a new maximum entropy summary tree S'. To get S', combine x as above with either its parent or a sibling, thereby decreasing the number of summary tree nodes by one, and then split off v_{i+1} from $other_v$ and create a node to represent $T_{v_{i+1}}$, thereby increasing the number of summary tree nodes back to k. Now, let s_0 denote the sum of the sizes of all the children of v in $other_v - \{v_{i+1}, v_j\}$. W times the increase in entropy from this two-part change to S is at least

$$\left[(s_0 + s_{v_j}) \lg \frac{1}{s_0 + s_{v_j}} + s_{v_{i+1}} \lg \frac{1}{s_{v_{i+1}}} - (s_0 + s_{v_{i+1}} + s_{v_j}) \lg \frac{1}{s_0 + s_{v_{i+1}} + s_{v_j}}\right] - s_{v_i}$$
$$= (s_0 + s_{v_j}) \lg \frac{s_0 + s_{v_{i+1}} + s_{v_j}}{s_0 + s_{v_j}} + s_{v_{i+1}} \lg \frac{s_0 + s_{v_{i+1}} + s_{v_j}}{s_{v_{i+1}}} - s_{v_i} \geq s_{v_{i+1}} - s_{v_i} \geq 0.$$

(The first inequality follows because $s_{v_j} \geq s_{v_{i+1}}$, which implies that $(s_0 + s_{v_{i+1}} + s_{v_j})/s_{v_{i+1}} \geq 2$.) But this is a nonnegative increase in entropy, proving that S' is a maximum entropy summary tree.

Splitting off v_{i+1} from $other_v$ decreases Φ by at least M^{n-d}, because the depth of the $other_v$ node equals the depth of node v_i, which is d. Hence the total $\Delta\Phi$ is at most $-M^{n-d} + 2n \cdot M^{n-d-1} = -M^{n-d}(1 - 2n/M) < 0$, a contradiction to the fact that S is a maximum entropy summary tree of minimum Φ. ∎

This completes the proof of Theorem 1. ∎

Theorem 2. *For all v, if $other_v$ exists, then $|other_v| \geq d_v - K + 2$.*

Proof. Each child of v not in $other_v$ contributes at least one node to the final summary tree, which has order $k \leq K$, and hence the number of children not in $other_v$ cannot exceed $K - 2$ (for one node is needed to represent $\{v\}$). ∎

3 The Exact Algorithm

Relabel the nodes as $1, 2, ..., n$, with the root being node 1, the nodes at depth d getting consecutive labels, and the children of a node being labeled with increasing consecutive labels in nondecreasing size order. (This can be done by processing the nodes in nondecreasing order by depth, with all the children of node v processed consecutively in nondecreasing order by size.) This relabeling costs $O(n \log n)$ time,[3] because $\sum_v (d_v \log d_v) \leq \sum_v (d_v \log n) \leq n \log n$.

The description and the implementation of the algorithm are simplified if we compute what we call the"pseudo-entropy," of summary trees for T_v rather than their entropy. The *pseudo-entropy p-ent*(S_v) of a tree S_v with nodes of weights $W_1, W_2, \ldots, W_k$ is simply $-\sum p_i \log p_i$, where $p_i = W_i/W$ and W is the weight of T (and *not* of T_v). Clearly, if S_v is part of a summary tree S for T, then S_v

[3] In fact, the relative order, at node v, of its $d_v - K + 1$ smallest-sized children does not matter since they must all be included in $other_v$. This allows us to perform just a partial sort at each node, in which the $d_v - K + 1$ smallest-size children are identified by selection and then the remaining at most $K - 1$ children are sorted. This improves the $O(n \log n)$ term to $O(n \log K)$ which is dominated by $O(nK)$.

contributes $-\sum p_i \log p_i$ to the entropy of S. Let $\text{ent}(S_v)$ denote the entropy of tree S_v. Then

$$\begin{aligned}\text{ent}(S_v) &= -\sum_i \frac{W_i}{W_v} \log \frac{W_i}{W_v} \ = \ -\left[\frac{W}{W_v} \sum_i \frac{W_i}{W} \log \frac{W_i}{W} + \sum_i W_i W \log \frac{W}{W_v}\right] \\ &= -\frac{W}{W_v} \text{p-ent}(S_v) - \log \frac{W}{W_v}.\end{aligned}$$

Thus the same tree optimizes the entropy and the pseudo-entropy.

We will be using a dynamic programming algorithm. To simplify the presentation we will only describe how to compute the maximum pseudo-entropy for a k-node summary tree for T_v, for each node v and for all k, $1 \le k \le \min\{K, n_v\}$.

The algorithm will first seek to find the value of the pseudo-entropy for optimal k-node summary trees when $other_v$ is restricted to being a prefix set, and then when $other_v$ is restricted to being a near-prefix set containing v_j as its non-prefix element, for each possible v_j in turn, i.e., for $\max\{3, d_v - K + 3\} \le j \le d_v$. Thus the algorithm will consider $\min\{d_v - 1, K - 1\} \min\{d_v, K - 1\}$ classes of candidate $other_v$ sets.

To describe the algorithm it will be helpful to introduce the notion of a summary forest. A k-node *summary forest for* T_v is a $(k+1)$-node summary tree for T_v from which v has been excised (leaving a forest). We will also call this a summary forest for $T_{v_1}, T_{v_2}, \ldots, T_{v_{d_v}}$. A summary forest for $T_{v_1}, T_{v_2}, \ldots, T_{v_l}$ is defined analogously, for $1 \le l \le d_v$.

To find the pseudo-entropy-optimal k-node summary trees for T_v, for $1 \le k \le K$, we first find the pseudo-entropy of optimal k-node summary forests for $T_{v_1}, T_{v_2}, \ldots, T_{v_l}$, for $\max\{1, d_v - K + 2\} \le l \le d_v$. The optimal k-node summary trees for T_v are then obtained by attaching $\{v\}$ as a root node to the trees in the optimal $(k-1)$-node summary forests for $T_{v_1}, T_{v_2}, \ldots, T_{v_{d_v}}$.

Now we explain how to find these optimal summary forests. In turn, we consider each of the up-to-$\max\{1, K-1\}$ possible classes of $other_v$ nodes: the prefix $other_v$ nodes, and for each j with $\max\{3, d_v - K + 3\} \le j \le d_v$, the class of near-prefix $other_v$ nodes including v_j as the non-prefix element.

First, we describe the handling of the candidate prefix $other_v$ nodes. We start with optimal k-node summary trees for T_{v_1}, for $1 \le k \le K - 1$. Inductively, suppose that we have computed (the entropy of) optimal k-node summary forests for $T_{v_1}, \ldots, T_{v_l}$. We find optimal k-node summary forests for $T_{v_1}, \ldots, T_{v_l}, T_{v_{l+1}}$ as follows. For $k = 1$, the forest comprises a single $other_v$ node. For each $k > 1$, we choose the highest entropy among the following options: an optimal h-node summary forest for $T_{v_1}, \ldots, T_{v_l}$ plus an optimal $(k-h)$-node summary tree for $T_{v_{l+1}}$, for $1 \le h < k$.

The correctness of this procedure is immediate: for $k = 1$ clearly the only summary forest is a one-node forest. For $k > 1$, $T_{v_{l+1}}$ cannot be represented by the $other_v$ node (since we are discussing the handling of the *prefix* $other_v$ nodes) and so it must be represented by one tree in the summary forest; this implies that $T_{v_1}, T_{v_2}, \ldots, T_{v_l}$ must also be represented by one or more trees in the summary forest. Of course, the representation of each of the parts must be optimal.

Our algorithm considers all possible ways of partitioning the nodes in the summary forest among these two parts; consequently it finds an optimal forest.

The process when v_j is the non-prefix node in $other_v$ is essentially identical. There are two changes: (i) $other_v$ is initialized to contain T_{v_j} (rather than .eing the empty set) and (ii) the incremental sweep skips tree T_{v_j}. The correctness argument is as in the previous paragraph.

Finally, to obtain optimal k-node summary forests for $T_{v_1}, T_{v_2}, \ldots, T_{v_{d_v}}$ one simply takes the best among the k-node forests computed for the different classes of candidate $other_v$ nodes. Again, correctness is immediate.

Theorem 3. *The running time of the algorithm is $O(K^2 n + n \log n)$.*

NOTE. Our time bound is $O(K^2 n + n \log n)$ to build K maximum-entropy summary trees, or $O(Kn + (n \log n)/K)$ amortized time for each. There is an obvious lower bound of $\Omega(n + K^2)$ to build all K trees, since one has to read an n-node tree and produce trees having $1, 2, 3, \ldots, K$ nodes. Hence there cannot be a $O(n)$-time algorithm that generates all K trees, since it would violate the lower bound when K is $\omega(\sqrt{n})$. Of course, conceivably there is a linear-time algorithm to build a maximum-entropy k-node summary tree for a single value of k.

Proof. The running time is the sum of three terms:
(1) $O(n \log n)$, for sorting the children of all nodes by size.
(2) $O(Kn)$ for initializations. In fact, the initializations for node v take time $O(K \cdot \min\{d_v, K-1\})$, which is $O(Kn)$ time in total.
(3) For each node v, the cost of processing node v_l when processing each of the classes of candidate $other_v$ nodes. Let $\langle v_a, v_{a+1}, \ldots, v_{v_d} \rangle$ be the sequence of nodes processed when considering the candidate prefix $other_v$ sets (nodes $v_1, \ldots, v_{a-1}$ are the nodes guaranteed to be in $other_v$). When processing the near-prefix candidate $other_v$ sets with non-prefix element v_j, the same sequence will be processed except that v_j will be omitted. For the class of prefix candidate sets, the cost for processing v_{l+1}, for $a \le l < v_d$, is $\min\{K-1, n_{v_a} + n_{v_{a+1}} + \cdots + n_{v_l}\} \cdot \min\{K-1, n_{v_{l+1}}\} \le \min\{K-1, n_{v_1} + n_{v_2} + \cdots + n_{v_l}\} \cdot \min\{K - 1, n_{v_{l+1}}\}$, for we are seeking k-node summary forests for $1 \le k \le K-1$, and the number of nodes in a summary tree cannot be more than the number of nodes available in the relevant subtrees of T. The same bound applies for each of the remaining classes of candidate $other_v$ sets and there are at most $K-1$ of these classes. Since the number of child nodes being processed when computing at node v is $d_v - a + 1 \le d_v$, the obvious upper bound here is $O(K^3 \cdot d_v)$. Summed over all v, this totals $O(K^3 \cdot n)$. However, Corollary 1 below shows that $\sum_{\text{non-leaf } v} \sum_l \min\{n_{v_1} + n_{v_2} + \cdots + n_{v_l}, K\} \cdot \min\{n_{v_{l+1}}, K\} \le 2Kn$, giving an overall time of $O(n \log n + K^2 n)$. ■

4 A Lemma for Running Time Analysis

In this section we state a lemma underlying the running time analysis of both the greedy algorithm and the exact algorithm. Let n be a positive integer and let T be a rooted, n-node tree, and *for this section only*, let $v_1, v_2, ..., v_{d_v}$ be v's children in *any* order.

Definition 3. *Relative to T, let cost(v) be defined for all $v \in T$ as follows. If v is a leaf, cost$(v) = 0$. If v is not a leaf, cost$(v) = [\sum_{i=1}^{d_v} cost(v_i)] + [\sum_{i=1}^{d_v-1} \min\{n_{v_1} + n_{v_2} + \cdots + n_{v_i}, K\} \cdot \min\{n_{v_{i+1}}, K\}]$.*

Lemma 4. [Proof omitted.] *For all v, cost$(v) \leq n_v^2$ if $n_v \leq K$, and cost$(v) \leq 2Kn_v - K^2$, if $n_v > K$.*

Corollary 1. [Proof omitted.] *For $K \geq 1$, $\sum_{non\text{-}leaf\ v}[\sum_{i=1}^{d_v-1} \min\{n_{v_1} + n_{v_2} + \cdots + n_{v_i}, K\} \cdot \min\{n_{v_{i+1}}, K\}] \leq 2Kn$.*

5 Greedy Algorithm

The greedy algorithm proposed in [4] is precisely the algorithm proposed herein for the exact solution but with the *other* sets restricted to being prefix sets. In [4] Greedy was shown to run in time $O(K^2 n + n \log n)$. Here, we shave off a factor of K from the first term.

Corollary 2. *(of Lemma 4).* [Proof omitted.] *The time needed by the greedy algorithm to generate summary trees of orders $k = 1, 2, \ldots, K$ is $O(Kn + n \log n)$.*

6 Improved Approximation Algorithm

In this section we describe an algorithm that computes an approximately entropy-optimal k-node summary tree. Our algorithm relies on the following outline from [4]:

1. One can rescale the weights in a tree to make them sum up to any positive integral value W_0, while leaving the entropy of any summary tree unchanged. (This is obvious.)
2. One can use algorithmic discrepancy theory to round each resulting real node weight w_v to value w'_v equal to either $\lfloor w_v \rfloor$ or $1 + \lfloor w_v \rfloor$ such that for each node $v \in T$, $|\sum_{u \in T_v} w'_u - \sum_{u \in T_v} w_u| \leq 1$ *for all v simultaneously*, without changing the overall sum.
3. Using Naudts's theorem [8] that almost identical probability distributions have almost identical entropy, one can prove, for some integer W_0 which is $O((K/\epsilon) \log(K/\epsilon))$, that if one finds a maximum entropy summary tree T^* for the modified weights (w'_v), then T^* has entropy (measured according to the *original* weights w_v) at most ϵ less than that of the truly maximum entropy summary tree.

Suppose that the weights on T are integral and sum to W_0. Clearly the number of nodes of positive weight cannot exceed W_0; however, the 0-weight nodes could far outnumber the positive-weight nodes. Indeed, that is exactly what happens if $n \gg W_0$.

Our algorithm exploits the fact that little processing is needed for most of the 0-weight nodes. In fact, we will need to compute summary trees for only the non-zero weight nodes and for at most $2(W_0 - 1)$ 0-weight nodes.

The algorithm works with a tree T', a reduced version of T in which some 0-weight nodes have been removed. The following notation will be helpful. $F_T(v, k)$ denotes the maximum pseudo-entropy of a k-node summary tree of T_v, where T_v is a subtree of tree T; similarly, $F_{T'}(v, k)$ denotes the maximum pseudo-entropy of a k-node summary tree of T'_v, where T'_v is a subtree of tree T'.

T' is obtained from T as follows: for each positively-sized node v in T, if v has one or more size-0 children, remove them and their descendants and replace them all by a single 0-weight child. Clearly optimal summary trees in T' form optimal summary trees in T (for the only difference in summarizing T is that we could add 0-weight nodes no longer present in T', and these would contribute 0 to the entropy). Note that if v is a 0-weight non-leaf node in T' then it must have non-zero size (assuming T has at least one positive-weight node). The following result is immediate.

Lemma 5. *Let T have n nodes and T' have n' nodes. Let v be a node in T' with $n(v)$ descendants in T and $n'(v)$ descendants in T'. Then $F_T(v, k) = F_{T'}(v, k)$ for $1 \le k \le n'(v)$. For $n'(v) + 1 \le k \le n$, $F_T(v, k) = F_{T'}(v, n'(v))$.*

Note that $F_{T'}(v, n'(v))$ is attained by a partition of the set of v's children in T' into singletons.

(Now of course we *have* changed the problem, since T' might have fewer than K nodes. However, if this happens, then optimal summary trees of T having more than $|T'|$ nodes have no more entropy than optimal summary trees of T having exactly $|T'|$ nodes.)

Even after the reduction it may be the case that $|T'| \gg W_0$, for T' might still contain long paths of 0-weight nodes in which each node has only one positively-sized child. However, the following lemmas show that they add little to the cost of computing optimal summary trees.

Lemma 6. *Let v be a 0-weight node in T' with a single child u. Then for $2 \le k \le |T'_v|$, $F_{T'}(v, k+1) = F_{T'}(u, k)$; also $F_{T'}(v, 1) = F_{T'}(u, 1)$.*

Proof. For $k \ge 2$, the $(k+1)$-node summary tree for T'_v adds a zero-weight node $\{v\}$ to the k-node summary tree for T'_u. For $k = 1$ both trees have a single node of weight w_u.

Lemma 7. *Let v be a 0-weight node in T' with exactly two children, a 0-weight leaf v_1 and a child u of positive size. Then for $3 \le k \le |T'_v|$, $F_{T'}(v, k+2) = F_{T'}(u, k)$; also $F_{T'}(v, 2) = F_{T'}(v, 1) = F_{T'}(u, 1)$.*

The proof of this lemma is essentially the same as that of Lemma 6. The following corollary is immediate.

Corollary 3. *Let $v_1, v_2, \ldots, v_l$, for $l > 1$, be a descending path of 0-weight nodes in T' such that each v_i, $1 \le i \le l$ either has one child, or has exactly two children* *one of which is a 0-weight leaf. Further suppose that l' of these nodes are in the second category. Node v_l must have a child of positive size (as otherwise $v_1 \neq v_l$ would be a size-0 non-leaf). Let u be the child of v_l of positive size. Then for $1 \leq k \leq |T'_{v_1}| - (l + l')$, $F_{T'}(v_1, k + l + l') = F_{T'}(u, k)$; and for $j \leq l + l'$, $F_{T'}(v_1, j) = F_{T'}(u, 1)$.*

This corollary implies that given the entropies of optimal entropy summary trees at a node u at the bottom of a maximal path of 0-weight nodes one can obtain the entropies of the optimal entropy summary trees at node v_1 at the top of the path in time $O(K)$.

At the remaining nodes in T' we perform the same computation as in the exact algorithm. As we can show, there are $O(W_0)$ such nodes, which leads to the following running time bound.

Theorem 4. [Proof omitted.] *The approximation algorithm to obtain a summary tree that has entropy within an additive ϵ of the optimal summary runs in time $O(n + W_0 \cdot K^3)$, where $W_0 = O((K/\epsilon)\log(K/\epsilon))$.*

Acknowledgments. We thank the referees for their helpful suggestions.

References

1. Beermann, D., Munzner, T., Humphreys, G.: Scalable, Robust Visualization of Very Large Trees. In: Proc. EuroVis, pp. 37–44 (2005)
2. Card, S.K., Nation, D.: Degree-of-Interest Trees: A Component of an Attention-Reactive User Interface. In: Proceedings of the Working Conference on Advanced Visual Interfaces, pp. 231–245 (2002)
3. Heer, J., Card, S.K.: DOI Trees Revisited: Scalable, Space-Constrained Visualization of Hierarchical Data. In: Advanced Visual Interfaces, pp. 421–424 (2004), `http://vis.stanford.edu/papers/doitrees-revisited`
4. Karloff, H., Shirley, K.: Maximum Entropy Summary Trees. In: Eurovis 2013 (2013), `www2.research.att.com/~kshirley/KarloffShirleyWebsite.pdf`
5. Lamping, J., Rao, R., Pirolli, P.: A Focus+Context Technique Based on Hyperbolic Geometry For Visualizing Large Hierarchies. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pp. 401–408 (1995), `http://dx.doi.org/10.1145/223904.223956`
6. von Landesberger, T., Kuijper, A., Schreck, T., Kohlhammer, J., van Wijk, J.J., Fekete, J.-D., Fellner, D.W.: Visual Analysis of Large Graphs: State-of-the-Art and Future Research Challenges. Computer Graphics Forum 30(6), 1719–1749 (2011)
7. Munzner, T., Guimbretiere, R., Tasiran, S., Zhang, L., Zhou, Y.: TreeJuxtaposer: Scalable tree comparison using Focus+Context with guaranteed visibility. ACM Transactions on Graphics 22(3), 453–462 (2003)
8. Naudts, J.: Continuity of a Class of Entropies and Relative Entropies. Reviews in Mathematical Physics 16(6), 809–822 (2004)
9. Shneiderman, B.: The Eyes Have It: A Task by Data Type Taxonomy for Information Visualization. In: Proceedings of the IEEE Symposium on Visual Languages, pp. 336–343 (1996)