Approximate Hotlink Assignment*

Evangelos Kranakis[1], Danny Krizanc[2], and Sunil Shende[3]

[1] Carleton University, School of Computer Science, Ottawa, ON, K1S 5B6, Canada.
e-mail: kranakis@scs.carleton.ca
[2] Department of Mathematics, Wesleyan University, Middletown CT 06459, USA.
e-mail: dkrizanc@wesleyan.edu
[3] Department of Computer Science, Rutgers University, Camden, NJ, 08102, USA.
e-mail: shende@crab.rutgers.edu

Abstract. Consider a directed rooted tree $T = (V, E)$ of maximal degree d representing a collection V of web pages connected via a set E of links all reachable from a source home page, represented by the root of T. Each leaf web page carries a weight representative of the frequency with which it is visited. By adding hotlinks, shortcuts from a node to one of its descendents, we are interested in minimizing the expected number of steps needed to visit the leaf pages from the home page. We give an $O(N^2)$ time algorithm for assigning hotlinks so that the expected number of steps to reach the leaves from the root of the tree is at most $\frac{H(p)}{\log(d+1)-(d/(d+1))\log d} + \frac{d+1}{d}$, where $H(p)$ is the entropy of the probability (frequency) distribution $p =< p_1, p_2, , \ldots, p_N >$ on the N leaves of the given tree, i.e., p_i is the weight on the ith leaf. The best known lower bound for this problem is $\frac{H(p)}{\log(d+1)}$. Thus our algorithm approximates the optimal hotlink assignment to within a constant for any fixed d.

1 Introduction

In an attempt to enhance the experience and reduce the latency of the average user, a number of authors have suggested ways of improving the design of websites, such as promoting and demoting pages, highlighting links, and clustering related pages in an adaptive fashion depending on user access patterns [4, 7]. In this paper we consider the strategy of adding "hotlinks", i.e., shortcuts from web pages at or near the home page of a site to popular pages a number of levels down in the (generally directed) network of pages. The idea of "hotlinks" was suggested by Perkowitz and Etzioni [7] and studied earlier by Bose et al. [2] for the special case of a website represented by a complete binary tree. Experimental results showing the validity of this approach are given in [3].

We model a website as a rooted directed tree $T = (V, E)$ where V is a collection of webpages connected by a set E of links. We assume that all webpages are reached starting from the root of the tree, representing the home page of the

* Research supported in part by NSERC (Natural Sciences and Engineering Research Council) of Canada and MITACS (Mathematics of Information Technology and Complex Systems) grants.

site and that users are interested in accessing information stored at the leaf web pages. Each leaf carries a weight representative of the frequency with which it is visited, i.e., its popularity. Our goal in adding hotlinks (directed edges from a node to one of its descendents) is to minimize the expected number of pages a user would visit when starting at the root and attempting to reach a leaf. We assume a user will always follow a hotlink (u, v) if after reaching u he or she wishes to reach a leaf that is a descendent of v. Note that this implies that adding hotlinks to a tree results in a new tree, not a general directed graph. We restrict ourselves to the case where at most one hotlink is added per node, but our results can be extended to the case where more than one hotlink can be added per node.

Consider a rooted directed tree T with N leaves and of maximal degree d. Let T^A be the tree resulting from an assignment A of hotlinks. The expected number of steps from the root to find a leaf web page is

$$E[T^A, p] = \sum_{i \text{ is a leaf}} d_A(i) p_i,$$

where $d_A(i)$ is the distance of the node i from the root in T^A, and $p =< p_i : i = 1, \ldots, N >$ is the probability distribution on the leaves of the original tree T. We are interested in finding an assignment A which minimizes $E[T^A, p]$.

An lower bound on $E[T^A, p]$ was given in [2] using information theory. Let $H(p)$ be the Entropy (see [1]) of the probability distribution $p =< p_i : i = 1, \ldots, N >$, which is defined by the formula[1]

$$H(p) = \sum_{i=1}^{N} p_i \log(1/p_i). \tag{1}$$

A tree of maximal degree d can be thought of as the encoding of the leaves with the d-symbol alphabet $0, 1, \ldots, d-1$. Adding a hotlink increments the alphabet by a single symbol to form an alphabet with $d+1$ symbols. Using this and the theory of prefix codes the following result can be proved (see [1, 2]):

Theorem 1 ([2]). *Consider an arbitrary rooted tree of maximal degree d. For any probability distribution $p =< p_1, p_2, \ldots, p_N >$ on the leaves of the tree and any assignment of at most one hotlink per node the expected number of steps to reach a web page located at a leaf from the root of the tree is at least $\frac{H(p)}{\log(d+1)}$.* ■

Our main result is to show that the above lower bound can be matched to within a constant for any constant d.

Theorem 2. *Consider an arbitrary rooted tree of maximal degree d. There is an algorithm, quadratic in the number of vertices of the tree, which for any probability distribution $p =< p_1, p_2, \ldots, p_N >$ on the leaves of the tree assigns one hotlink per node in such a way that the expected number of steps to reach a leaf of the tree from the root is at most $\frac{H(p)}{\log(d+1) - (d/(d+1)) \log d} + \frac{d+1}{d}$.* ■

[1] Throughout this paper log denotes logarithm in base 2 and ln logarithm in base e.

Section 2 provides the proof of the main theorem for the case of binary trees. Subsection 2.1 provides the proof of a useful lemma concerning entropies. In Section 3 we extend the proof to arbitrary trees of maximum degree d. In Section 4 we discuss an improved analysis of our algorithm on complete trees and in Section 5 we conclude with some open problems.

2 Hotlink Assignments for Binary Trees

In this section we give the hotlink assignment that achieves the upper bound of the main theorem. For the sake of simplicity we first consider the case of binary trees. Later we adapt the result to the case of trees of maximum degree d.

2.1 A useful lemma on entropies

Before giving the hotlink assignment algorithm and its analysis we present a useful lemma concerning entropy.

Consider a probability distribution $p =< p_1, p_2, \ldots, p_N >$ and a partition $A_1, A_2, \ldots, A_k$ of the index set $\{1, 2, \ldots, N\}$ into k non-empty subsets. Define

$$S_i = \sum_{j \in A_i} p_j, \text{ for } i = 1, 2, \ldots, k. \tag{2}$$

Consider the new distributions:

$$p^{(i)} =< p_j^{(i)} := \frac{p_j}{S_i} : j \in A_i >, \text{ for } i = 1, 2, \ldots, k. \tag{3}$$

Lemma 1. *For any partition $A_1, A_2, \ldots, A_k$ of the index set of the probability distribution we have the identity*

$$H(p) = \sum_{i=1}^{k} S_i H(p^{(i)}) - \sum_{i=1}^{k} S_i \log S_i, \tag{4}$$

where S_i and $p^{(i)}$ are defined in Equations (2, 3).

Proof The proof is a straightforward application of the definition of the entropy. We have

$$\begin{aligned} H(p) &= \sum_{j=1}^{N} -p_j \log p_j \\ &= \sum_{i=1}^{k} - \sum_{j \in A_i} p_j \log p_j \\ &= \sum_{i=1}^{k} -S_i \sum_{j \in A_i} p_j^{(i)} \log p_j \end{aligned}$$

$$= \sum_{i=1}^{k} -S_i \sum_{j \in A_i} p_j^{(i)} (\log p_j^{(i)} + \log S_i)$$

$$= \sum_{i=1}^{k} S_i H(p^{(i)}) - \sum_{i=1}^{k} S_i \log S_i,$$

which proves the Lemma. ∎

2.2 Algorithm for binary trees

Before we proceed any further we need to show how to assign weights to all the nodes of the tree. In $O(N)$ time we can propagate the original weights on the leaves of the tree through the entire tree using a bottom-up process. This is done inductively as follows. The weight of the ith leaf is equal to p_i. The weight of a node u is equal to the sum of the weights of its children. Finally, we define the weight of a subtree to be equal to the weight of the root of this subtree. We present the following well-known lemma for completeness.

Lemma 2. *Consider a probability distribution $p =< p_1, p_2, \ldots, p_N >$ on the N leaves of a binary tree. Then either there is a tree node whose weight is between $1/3$ and $2/3$ or else there is a leaf whose weight is $> 2/3$.*

PROOF Assume there is no tree node whose weight is in the range $[1/3, 2/3]$. We will show that there is a leaf whose weight is $> 2/3$. Start from the root. All its descendants have weight outside the range $[1/3, 2/3]$. Take a descendant c that has weight $> 2/3$. If this is a leaf then we are done. So assume it is not a leaf. All descendants of c have weight outside the range $[1/3, 2/3]$. Hence, c has a descendant whose weight is $> 2/3$. If this is a leaf then we are done. So assume it is not a leaf. Iterate this procedure and we see that there must exist a leaf of weight $> 2/3$. This completes the proof of the Lemma. ∎

Theorem 3. *Consider a rooted binary tree. There is an algorithm, quadratic in the number of vertices of the tree, which for any probability distribution on the leaves of the tree assigns a hotlink per node in such a way that the expected number of steps to reach a leaf of the tree is at most $aH(p)+b$, where $a = \frac{1}{\log 3 - 2/3}$ and $b = \frac{3}{2}$.*

PROOF As mentioned before in $O(N)$ time we can propagate the original weights on the leaves of the tree through the entire tree. Once all these internal node weights are assigned we use a top-down method to assign hotlinks.

The assignment of hotlinks is done recursively. We partition the tree T into three subtrees T_1, T_2, T_3. The partition is determined as follows. Find a node u that determines a subtree T_2 rooted at u such that the weight of u is bounded from below by $1/3$ and from above by $2/3$. I.e.,

$$\frac{1}{3} \cdot weight(T) \leq weight(u) \leq \frac{2}{3} \cdot weight(T). \tag{5}$$

If u is not a child of c then do the following. Without loss of generality assume T_2 is contained entirely inside the subtree rooted at the left child of the root. Then T_1 is the tree rooted at the left child of the root minus the tree T_2 and T_3 is the tree rooted at the right child of the root. The recursive process assigns a hotlink to the root u of the subtree T_2. If however, the only node u satisfying Inequalities (5) must be a child of c then we select u to be the heaviest grandchild of c, which must have weight at least 1/4 of the weight of T (this is because the tree is binary). If no such node exists then we choose for u the heaviest leaf of the tree which is guaranteed to have weight greater than 2/3. The recursive process assigns a hotlink to this new node u. The trees T_1, T_2, T_3 are defined exactly as before, and moreover none of them has weight bigger than 2/3 of the weight of T. Then we recursively assign hotlinks to the subtrees T_1, T_2, T_3. The idea is illustrated in Figure 1.

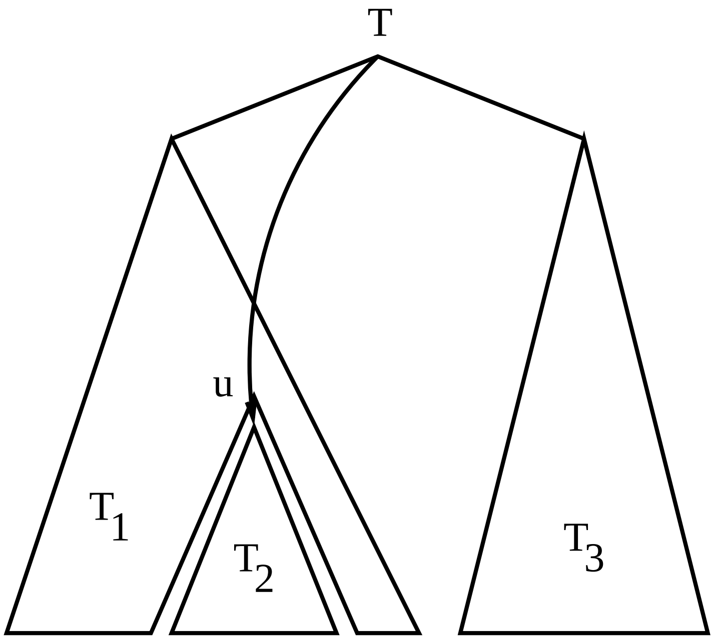

Fig. 1. Assignment of hotlinks: we assign a hotlink from the root to a "heavy" node u and iterate the recursive assignment to the subtrees T_1, T_2, T_3.

The precise algorithm **HotlinkAssign** which determines the hotlink assignment is defined as follows.

HotlinkAssign(T)
Initialize: $c :=$ root of T, $l :=$ left ($r :=$ right) child of c ;

if c has grandchildren **do**;
  1a. **find** u descendant of c such that
      $weight(T)/3 \leq weight(u) \leq 2weight(T)/3$
  1b. if no such descendant exists let u be a max weight leaf;
  2a. **if** distance from c to u is ≥ 2
      then add a hotlink from c to u
  2b. **else** let u be the (any) grandchild of c with heaviest weight;
      add a hotlink from c to u
  3. $T_2 :=$ tree rooted at u;
  4. Let v ancestor of u that is child of c;
     w.l.o.g. assume $v = l$;
  5. $T_1 :=$ tree rooted at left child of c minus T_2;
  6. $T_3 :=$ tree rooted at right child of c;
  7. **HotlinkAssign** (T_1),
  8. **HotlinkAssign** (T_2),
  9. **HotlinkAssign** (T_3)
end;

We will prove by induction on the depth of the tree T that there exist constants a, b such that for the hotlink assignment A described above

$$E[T^A, p] \leq aH(p) + b.$$

It will become apparent from the proof below how to select a and b (see Inequalities (6) and (7)).

The initial step, when the depth of the tree is 1, is trivial because we will choose b so that $b \geq 1$. Assume the induction hypothesis is valid for the subtrees of T. We calculate costs of the resulting hotlink assignments. According to Lemma 2 the hotlink assigned from the root to a node partitions the leaves of the tree into three subsets A_1, A_2, A_3 with corresponding weights S_1, S_2, S_3. If Lemma 2 chooses a node that has weight between 1/3 and 2/3 (or if it is the child of the root and a grandchild is chosen with weight greater than or equal to 1/4) then it is easy to see that all three S_i's have weight $\leq 2/3$. If Lemma 2 chooses a leaf then $S_2 > 2/3$ and $S_1 + S_3 < 1/3$.

In the first case, using the notation of Lemma 1 we obtain

$$\begin{aligned}
E[T^A, p] &= \sum_{i=1}^{3} S_i(1 + E[T_i^A, p^{(i)}]) \\
&= 1 + \sum_{i=1}^{3} S_i E[T_i^A, p^{(i)}] \\
&\leq 1 + \sum_{i=1}^{3} S_i(aH(p^{(i)}) + b) \\
&= 1 + b + a\sum_{i=1}^{3} S_i H(p^{(i)})
\end{aligned}$$

$$
\begin{aligned}
&= 1 + b + aH(p) + a\sum_{i=1}^{3} S_i \log S_i \\
&\leq aH(p) + b.
\end{aligned}
$$

The last inequality being valid because we will choose the constant a such that

$$
1 + a\sum_{i=1}^{3} S_i \log S_i \leq 0. \tag{6}
$$

If A_2 is a leaf of weight greater than $2/3$ then T_2 is the empty tree and using the notation of Lemma 1 we obtain

$$
\begin{aligned}
E[T^A, p] &= S_2 + \sum_{i=1,3} S_i(1 + E[T_i^A, p^{(i)}]) \\
&= 1 + \sum_{i=1,3} S_i E[T_i^A, p^{(i)}] \\
&\leq 1 + \sum_{i=1,3} S_i(aH(p^{(i)}) + b) \\
&= 1 + a\sum_{i=1,2,3} S_i H(p^{(i)}) + (S_1 + S_3)b \\
&= 1 + aH(p) + a\sum_{i=1}^{3} S_i \log S_i + (S_1 + S_3)b \\
&\leq 1 + aH(p) + (S_1 + S_3)b \\
&\leq 1 + aH(p) + (1/3)b \\
&\leq aH(p) + b
\end{aligned}
$$

The last inequality being valid because we will choose the constant b such that

$$
1 + (1/3)b \leq b. \tag{7}
$$

We now consider the value of the constants a and b. Clearly, $b = 3/2$ satisfies Inequality (7). To choose a we first prove the following lemma.

Lemma 3. *The solutions of the optimization problem*

$$
\begin{aligned}
&\textit{maximize} \quad f(x,y) = x\ln x + y\ln y + (1-x-y)\ln(1-x-y) \\
&\textit{subject to} \quad 0 \leq x, y \leq 2/3, 1/3 \leq x + y \leq 1
\end{aligned}
$$

are such that $\{x, y, 1-x-y\} = \{0, 1/3, 2/3\}$.

PROOF The partial derivative of f with respect to x is equal to $\ln x - \ln(1-x-y)$, which is increasing as a function of x. This means that the maximum values of f (as a function only of x parametrized with respect to y) are obtained at the

endpoints of the interval on which it is defined, i.e., $\max\{0, 1/3 - y\} \leq x \leq \min\{2/3, 1 - y\}$. This gives rise to two cases depending on the value of y.
Case 1: $y \geq 1/3$.
In this case we have that the endpoints are $0 \leq x \leq 1 - y$ and the value of f at the endpoints is equal to

$$\begin{array}{c} f(0, y) = f(1 - y, y) = y \ln y + (1 - y) \ln(1 - y) \\ \text{subject to } 1/3 \leq y \leq 2/3. \end{array} \tag{8}$$

Case 2: $y \leq 1/3$.
In this case we have that the endpoints are $1/3 - y \leq x \leq 2/3$ and the value of f at the endpoints is equal to

$$\begin{array}{c} f(1/3 - y, y) = f(2/3, y) = (1/3 - y) \ln(1/3 - y) + y \ln y + (2/3) \ln(2/3) \\ \text{subject to } 0 \leq y \leq 1/3. \end{array} \tag{9}$$

In particular, the maximum value of f is attained at the maximum values of Cases 1 and 2 above. The functions in Equations (8, 9) depend only on the variable y and their maxima are obtained at the endpoints of the interval for y on which the function is defined. In Case 1, this is $1/3 \leq y \leq 2/3$ and in Case 2, this is $0 \leq y \leq 1/3$. It follows that for Case 1 we have that our function obtains its maximum value when $y = 1/3, 2/3$ and in Case 2 when $y = 0, 1/3$, i.e., $y = 0, 1/3, 2/3$. Consequently, when $y = 0$ we have $x = 1/3, 2/3$, when $y = 1/3$ we have $x = 0, 2/3$, and when $y = 2/3$ we have $x = 0, 1/3$. This proves the lemma. ■

Lemma 3 implies that $\sum_{i=1}^{3} S_i \log S_i \leq (2/3)\log(2/3) + (1/3)\log(1/3)$. Hence, Inequality (6) is satisfied when $a = \frac{1}{\log 3 - 2/3}$.

Concerning the running time of the algorithm we note that in linear time we can assign weights to the nodes of the tree bottom up. For each recursive call of the algorithm **HotlinkAssign** we must update the weights. Since the number of recursive calls does not exceed the height of the tree, we see that the running time is worst-case quadratic. The proof of Theorem 3 is now complete. ■

3 Trees of Maximum Degree d

Lemma 2 has a natural generalization for the case of trees with maximum degree d. Namely the following result is known.

Lemma 4. *Consider a probability distribution $p =< p_1, p_2, \ldots, p_N >$ on the N leaves of a tree of maximum degree d. Then either there is a tree node whose weight is between $1/(d+1)$ and $d/(d+1)$ or else there is a leaf whose weight is $> d/(d+1)$.* ■

3.1 Algorithm for trees of maximum degree d

Now we can prove our main result in Theorem 2. In the sequel we indicate only the necessary changes.

Proof (Outline) of Theorem 2. As with Theorem 3 we assign weights to all nodes of the tree in a bottom-up fashion. The assignment of hotlinks is done recursively in a top-down fashion. Let c be the root of T. Indeed, we partition the tree T into at most $d+1$ subtrees as follows. By Lemma 4, find a node u that determines a subtree T_i' rooted at u such that the sum of the weights of the leaves of T_i' is bounded from below by $1/(d+1)$ and from above by $d/(d+1)$. Without loss of generality assume this tree is a descendant of the ith child of the root. Then T_i is defined to be the tree rooted at the ith child of the root minus the tree T_i'. Also T_j, for $j \neq i$, is defined to be the tree rooted at the j-th child of the root. Now, T_i' and the sequence $T_1, T_2, \ldots, T_i, \ldots$ is the desired partition of the subtrees. As before, if only children of c are the only nodes whose weight is bounded from below by $1/(d+1)$ and from above by $d/(d+1)$ then we select u to be the (any) heaviest grandchild of c. The recursive process assigns a hotlink from c to the root u of the subtree T_i'. Then we recursively assign hotlinks to the subtrees $T_1, T_2, \ldots, T_d$.

Using Lemma 4 and an analysis similar to that of Theorem 3 we obtain the main result. As before we prove by induction on the depth of the tree T that there exist constants a, b such that for the hotlink assignment A described above

$$E[T^A, p] \leq aH(p) + b.$$

Inequality (6) is transformed into

$$1 + a \sum_{i=1}^{d+1} S_i \log S_i \leq 0, \tag{10}$$

and Inequality (7) into

$$1 + (1/(d+1))b \leq b. \tag{11}$$

Here, a and b are selected so as to satisfy Inequalities (10) and (11). The value of a follows immediately from the following lemma.

Lemma 5. *The solutions of the optimization problem*

$$\textit{maximize} \;\; f(s_1, s_2, \ldots, s_d) = \sum_{i=1}^{d} s_i \ln s_i + \left(1 - \sum_{i=1}^{d} s_i\right) \ln \left(1 - \sum_{i=1}^{d} s_i\right)$$

$$\textit{subject to} \;\; 0 \leq s_1, s_2, \ldots, s_d \leq \frac{d}{d+1}, \frac{1}{d+1} \leq \sum_{i=1}^{d} s_i \leq 1$$

are obtained when one among the quantities $s_1, \ldots, s_d, 1 - \sum_{i=1}^{d} s_i$ *attains the value* $d/(d+1)$*, another the value* $1/(d+1)$ *and all the rest are equal to* 0.

Proof The proof is by induction on the number of variables. For $d = 2$ this is just Lemma 3. Assume the lemma is true for $d - 1 \geq 2$. We will prove it for d. Set $x := s_d$ and $y := s_1 + \cdots + s_{d-1}$. The partial derivative of f with respect to x is equal to $\ln x - \ln(1 - x - y)$, which is increasing as a function of x. This

means that the maximum values of f (as a function only of x parametrized with respect to $s_1, \ldots, s_{d-1}$) are obtained at the endpoints of the interval on which it is defined, i.e., $\max\{0, 1/(d+1) - y\} \le x \le \min\{d/(d+1), 1-y\}$. This gives rise to two cases depending on the value of y.
Case 1: $y \ge 1/(d+1)$.
In this case we have that the endpoints are $0 \le x \le 1 - y$ and the value of f at the endpoints is equal to

$$\begin{aligned} f(s_1, \ldots, s_{d-1}, 0) &= f(s_1, \ldots, s_{d-1}, 1-y) \\ &= (1-y)\ln(1-y) \\ &\quad + s_1 \ln s_1 + \cdots + s_{d-1} \ln s_{d-1} \\ &\text{subject to } 1/(d+1) \le y \le 1, \text{ and } s_1, \ldots, s_{d-1} \le d/(d+1). \end{aligned} \tag{12}$$

Case 2: $y \le 1/(d+1)$.
In this case we have that the endpoints are $1/(d+1) - y \le x \le d/(d+1)$ and the value of f at the endpoints is equal to

$$\begin{aligned} f(s_1, \ldots, s_{d-1}, 1/(d+1) - y) &= f(s_1, \ldots, s_{d-1}, d/(d+1)) \\ &= (1/(d+1) - y)\ln(1/(d+1) - y) \\ &\quad + s_1 \ln s_1 + \cdots + s_d \ln s_{d-1} \\ &\quad + (d/(d+1))\ln(d/(d+1)) \\ &\text{subject to } 0 \le y \le 1/(d+1). \end{aligned} \tag{13}$$

Thus, we have reduced the original optimization problem to the two optimization problems described in Problems (12, 13) which have only $d-1$ variables (i.e., one variable less). It is trivial that in Problem (13) the optimal solution is obtained when $s_1 = \ldots = s_{d-1} = 0$. In this case $s_d = 1/(d+1)$ or $s_d = d/(d+1)$ and the inductive hypothesis is valid for d. In Case 1 we reduce to the same optimization problem on $d-1$ variables. Hence, by the induction hypothesis the optimal solutions are obtained when one among the quantities $s_1, \ldots, s_{d-1}, 1 - \sum_{i=1}^{d-1} s_i$ attains the value $d/(d+1)$, another the value $1/(d+1)$ and all the rest are equal to 0. Hence the result follows for d variables because in this case $s_d = 0$ or $s_d = 1 - \sum_{i=1}^{d-1} s_i$. The proof of the lemma is now complete. ∎

The rest of the details of the proof can now be left to the reader. The proof of Theorem 2 is complete. ∎

4 Analysis for Special Trees and Distributions

Our present analysis of the hotlink assignment problem focused on using the entropy of the distribution in order to bound the expected number of steps. In fact, our analysis still leaves a gap between the lower bound of Theorem 1 and the upper bound of Theorem 2. Can we improve the upper bound any further?

In the sequel we indicate that our algorithm still performs close to the lower bound for the uniform distribution on complete trees of degree d. We also indicate how to adapt our algorithm **HotlinkAssign** in the case of arbitrary distributions on complete trees.

First, consider the uniform distribution p on the leaves of the complete d-ary tree with $N \approx d^n$ leaves. The entropy of this distribution is $H(p) = n \log d$. Theorem 1 implies that

$$\frac{H(p)}{\log(d+1)} = \frac{\log d}{\log(d+1)} \cdot n \tag{14}$$

is a lower bound, while Theorem 2 implies that

$$\frac{H(p)}{\log(d+1) - (d/(d+1)) \log d} + \frac{d+1}{d} = \frac{\log d}{\log(d+1) - (d/(d+1)) \log d} \cdot n + \frac{d+1}{d}$$

is an upper bound on the expected number of steps.

However, in this case it is easy to see that the **HotlinkAssign** algorithm always picks a hotlink that is a grandchild of the current root. This observation can be used to give a different analysis of the algorithm. Using the method employed in [2][Theorem 3] we can show directly that the expected number of steps to reach a leaf is at most

$$\left(1 - \frac{1}{d^2}\right) \cdot n \tag{15}$$

plus an additive constant.

More generally, on a complete tree of degree d with an arbitrary distribution on the leaves we can change our algorithm **HotlinkAssign** so that a hotlink is placed always at the heaviest grandchild of the current root, i.e., we omit step 2a in algorithm **HotlinkAssign**. The analysis employed in [2][Theorem 3] as well as the resulting upper bound given in (15) is still valid. Moreover, it is easy to see that the lower bound in (14) and the upper bound in (15) are asymptotically identical, as $d \to \infty$.

5 Conclusions and Open Problems

In this paper we have considered the problem of assigning hotlinks to the nodes of a tree so as to minimize the expected number of steps from the root to the leaves under an arbitrary probability distribution. Our main result is an approximation algorithm for the case of bounded degree trees. A significant gap remains between the upper and lower bounds and further improvements would be of interest. It is expected that experimental results like the ones in [3] will provide additional insight on this problem. While it is known that the problem is NP-complete for DAGs, the complexity of the case of trees is still open. In this paper, we restricted ourselves to at most one hotlink added per node. Fuhrmann et al. [5] report results on the case of adding k links per node to a d-regular complete tree. Our results can be extended to the case of a fixed number, k, added per node (using known generalizations of lemma 2) but the gap between upper and lower bounds increases with k. Perhaps another approach will not suffer from this weakness. The variation where the total number of hotlinks does not exceed a certain fixed budget could be explored. Additional interesting problems include finding further improvements for special distributions, such as Zipf's distribution (e.g., see [8, 6]) which is especially relevant to this application.

References

1. N. Abramson, "Information Theory and Coding", McGraw Hill, 1963.
2. P. Bose, J. Czyzowicz, L. Gasienicz, E. Kranakis, D. Krizanc, A. Pelc, M. Vargas Martin, "Strategies for Hotlink Assignments," in Proceedings of ISAAC 2000, held in Taipei, Taiwan, Dec. 2000, D. T. Lee and Shang-Hua Teng, eds., SVLNCS, Vol. 1969, pp. 23-34.
3. J. Czyzowicz, E. Kranakis, D. Krizanc, A. Pelc, and M. Vargas Martin, "Evaluation of Hotlink Assignment Heuristics for Improving Web Access," in proceedings of the 2nd International Conference on Internet Computing (IC'2001), June 25th - 28th, 2001, Monte Carlo Resort, Las Vegas, Nevada, USA.
4. M. C. Drott, "Using Web Server Logs to Improve Site Design," Proceedings of ACM Conference on Computer Documentation, 1998, pp. 43-50.
5. S. Fuhrmann, S. O. Krumke and H.-C. Wirth, "Multiple Hotlink Assignment," Proceedings of Workshop on Graph-Theoretic Concepts in Computer Science, 2001, SVLNCS, to appear.
6. D. Knuth, "The Art of Computer Programming: Volume 3, Sorting and Searching", Addison Wesley, 3rd ed., 1998.
7. M. Perkowitz and O. Etzioni, "Towards Adaptive Web Sites: Conceptual Framework and Case Study", Proceedings of Eight WWW Conference, 1999.
8. C. K. Zipf. "Human Behavior and the Principle of Least Effort", Reading Mass., Addison Wesley, 1949.