PUC

ISSN 0103-9741

Monografias em Ciência da Computação

n° 27/07

# Improved Approximations for the Hotlink Assignment Problem and Binary Searching on Trees

**Eduardo Sany Laber**

**Marco Serpa Molinaro**

Departamento de Informática

**PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO DE JANEIRO**

**RUA MARQUÊS DE SÃO VICENTE, 225 - CEP 22451-900**

**RIO DE JANEIRO - BRASIL**

Monografias em Ciência da Computação, No. 27/07 ISSN: 0103-9741
Editor: Prof. Carlos José Pereira de Lucena December, 2007

# Improved Approximations for the Hotlink Assignment Problem and Binary Searching on Trees

**Eduardo Sany Laber and Marco Serpa Molinaro**

{laber,mmolinaro}@inf.puc-rio.br

**Abstract.** Let $G = (V, E)$ be a directed acyclic graph representing a web site, where nodes correspond to pages and arcs to hyperlinks. In this context, hotlinks are defined as shortcuts (new arcs) added to web pages of $G$ in order to reduce the time spent by users to reach their desired information.

In this paper, we consider the problem where $G$ is a rooted directed tree and the goal is minimizing the expected time spent by users by assigning at most $k$ hotlinks to each node. For the most studied version of this problem where at most one hotlink can be assigned from each node, we prove the existence of an FPTAS, improving upon the constant factor algorithm recently obtained in [Jacobs, WADS 2007]. In addition, we develop the first constant factor approximation algorithm for the most general version where $k$ hotlinks can be assigned from each node. Finally, we give an evidence that the technique developed to obtain this last result can be of independent interest since, based on it, we develop a linear time algorithm that provides the first constant approximation for the average case version of the problem of binary searching in trees, a natural generalization of the classical problem of searching in lists with different access probabilities.

**Keywords:** Approximation Algorithms, trees, searching.

**Resumo.** Seja $G = (V, E)$ um dígrafo acíclico representando um *web site*, onde nós correspondem a páginas e arcos a hiperlinks. Nesse contexto, hotlinks são definidos como atalhos (novos arcos) adicionados à páginas da internet para reduzir o tempo gasto por usuários para alcançar a informação desejada.

Nesse artigo nós consideramnos o problema onde $G$ é uma árvore direcionada enraizada e o objetivo é minimizar o tempo esperado gasto por usuários atribuindo no máximo $k$ hotlinks a cada nó. Para a versão mais estudada desse problema onde no máximo um hotlink pode ser adicionado a cada nó, nós provamos a existência de um FPTAS, melhorando sobre o algoritmo com garantia constante obtido recentemente em [Jacobs, WADS 2007]. Além disso, desenvolvemos o primeiro algoritmo com aproximação constante para a versão mais geral onde $k$ hotlinks podem ser adicionados a cada nó. Por fim, apresentamos uma evidência que a técnica desenvolvida para obter este último resultado pode ser de interesse independente, pois baseado nela desenvolvemos um algoritmo em tempo linear que obtém a primeira aproximação de fator constante para a versão de caso médio do problema de busca binária em árvores, uma generalização natural do clássico problema de busca em listas com diferentes probabilidades de acesso.

**Palavras-chave:** Algoritmos Aproximativos, árvores, busca.

**In charge of publications:**
Rosane Teles Lins Castilho
Assessoria de Biblioteca, Documentação e Informação
PUC-Rio Departamento de Informática
Rua Marquês de São Vicente, 225 - Gávea
22451-900 Rio de Janeiro RJ Brasil
Tel. +55 21 3527-1516 Fax: +55 21 3527-1530
E-mail: bib-di@inf.puc-rio.br
Web site: http://bib-di.inf.puc-rio.br/techreports/

## 1 Introduction

The huge growth of WWW has brought many new and interesting challenges to computer scientists. The investigation of several algorithmic problems related to search, classification and organization of information that could not be well motivated before WWW age are, nowadays, central to its good behavior. Among these problems, one that has attracted the attention of some people in the TCS community is the problem of optimizing user access in Web Sites. This problem can be addressed in different ways, which includes increasing the bandwidth of the site, maintaining copies of content in different servers and enhancing the site's navigational structure. Here we are interested in the latter approach.

On one hand, the navigational structure of a Web Site (its pages and its links) is designed in a way to be meaningful and helpful to users. On the other hand, it is not likely that the structure takes into account the fact that some information are much more sought than others. In fact, it may happen that a very 'popular' information is located much farther from the home page than a 'non popular' one. Then, a reasonable approach to optimize the access in a Web Site is enhancing its navigational structure through the addition of a set of shortcuts (hotlinks). This keeps the original structure untouched and allows reducing the expected length of the path from the home page to the desired information. In the implementation of this approach, the number of added shortcuts per page shall be small, otherwise pages may become polluted and disturb the navigation process. This scenario leads to the following algorithmic problem.

**Problem Definition.** Let $G = (V, E)$ be a DAG with $n$ nodes and a unique root $r$, and let $w : V \to \mathbb{Q}^+$ be a weight function. The graph $G$ models the site and $w(v)$, for each $v \in V$, the popularity of a page $v$. A $k$-hotlink assignment ($k$-assignment for short) $A$ for $G$ is a set of directed arcs that satisfies the following properties: (i) both endpoints of arcs in $A$ belong to $V$; (ii) for each node $u \in V$ there can be at most $k$ hotlinks of $A$ leaving $u$.

The cost of an assignment $A$ is given by $\mathrm{EP}(G, A, w) = \sum_{u \in V} d(r, u, G+A)w(u)$, where $d(r, u, G + A)$ is the length of the path traversed by a typical user (this will be detailed soon) from $r$ to $u$ in the enhanced graph $G+A = (V, E \cup A)$. An optimal assignment $A^*$ is one that minimizes $\mathrm{EP}(G, A, w)$ over all possible assignments $A$. Given a DAG $G$, with an unique source $r$, and a weight function $w : V \to \mathbb{Q}^+$, the $k-$Hotlink Assignment Problem ($k-$HAP for short) consists of finding an optimal $k$-hotlink assignment for $(G, w)$.

In this paper, we focus in the case where $G$ is a directed tree $T$ and the desired information is always on the leaves of $T$, that is, $w(u) = 0$ for every node $u$ that is not a leaf of $T$ (the case with weights on nodes can be modeled creating artificial leaves). As for the definition of distance $d(\cdot)$, the cost spent by a typical user to find his (her) target information is directly related to how he (she) navigates in the site. Two models of navigation have been considered in the literature: the clairvoyant user model and the greedy user model. The former is somehow unrealistic since it assumes that an user has a map of the entire site so that he (she) always knows how to follow a shortest path from the root to the target information. The latter assumes that the user always follows the link (original link or hotlink) that leads him (her) closest *in the original tree* $T$ to his (her) target information (Figure 1)

Like most of the papers in this subject, here we assume the greedy user model. The greedy assumption, together with the fact that $T$ is a directed tree, implies that only hotlinks from a node to its descendants can be followed by users. Thus, we can assume w.l.o.g. that every hotlink point from a node to one of its descendants in $T$.

**Related Work.** The idea of hotlinks was first suggested by Perkowitz and Etzioni [22]. In [5], Czyzowicz et al. present experimental results showing the validity of the hotlink approach. In addition, they describe a software tool to automatically assign hotlinks to web sites. Experimental results also appear in [24].

Turning our attention to theoretical results, in [3] Bose et. al. prove that the hotlink assignment problem is NP-Complete for DAG's in the clairvoyant user model. In addition, they use Shannon's coding theorem to prove that given a tree $T$ and a normalized weight function $w$, then $\mathrm{EP}(T, A, w) \geq H(w)/(\log(\Delta+1))$, for every 1-hotlink assignment $A$ for $T$, where $\Delta$ is the maximum degree of the input tree and $H(w) = -\sum_{u \in T} w(u) \log w(u)$ is the entropy induced by $w$.

In [16], Kranakis et. al. present a quadratic time algorithm that produces a 1-hotlink assignment $A$ such that $\mathrm{EP}(T, A, w) \leq \frac{H(w)\Delta}{\log \Delta}$ (for large $\Delta$). In [7] and [8], Douieb and Langerman present algorithms that construct 1-assignments whose associated costs are $O(H(w))$. This upper bound together with the above entropy lower bound guarantee that these methods provide a $O(\log n)$ approximation for the 1-HAP. In [8], it is also presented a way to construct a $k$-assignment with cost $O(H(w)/\log k)$. The first algorithm with constant approximation ratio for the 1-HAP is due to Jacobs [12] – it runs in $O(n^4)$ and achieves 2-approximation. In this same paper, Jacobs mentions that it is not clear how to extend his method to guarantee a constant approximation for the $k$-HAP.

Exact algorithms for the 1-HAP were independently discovered by Gerstel at. al. [10] and Pessoa et. al. [24] (see also [17] for a journal version merging both papers). The algorithm of [10] is exponential in the height of the input tree. Now notice that the paths that users take to reach the desired information induce a tree on $T + A$ (see Figure 1.c). We denote such tree by $T^A$ and refer to it as user tree. The algorithm of [24], which can be viewed as an optimized version of the one proposed in [10], has the following property: for each integer $D$, it calculates in $O(n2^D)$ the best 1-assignment among the 1-assignments $A$ such that the height of $T^A$ is at most $D$.

Variants and applications of the hotlink assignment problem have also been considered [4, 19, 23]. In [4], Bose et. al. discuss the use of hotlink assignments in asymmetric communication protocols [1] to achieve better performance bounds. The *gain* of a hotlink assignment $A$ is defined as the difference between the cost of the empty assignment and that of assignment $A$. Matichin and Peleg proposed a polynomial time algorithm that guarantees a constant approximation w.r.t. the maximum gain for DAG's [19]. In [12], Jacobs proposes a PTAS for approximating the maximum gain in trees. It shall be observed, however, that a constant approximation with respect to the gain may represent a linear approximation gap with respect to the expected path length considered here. In addition, we feel that the approximation in terms of the gain does not necessarily reflect the quality of the assignment. As an example, a 0.9 approximation for the gain (which is supposed to be a good approximation) may correspond to an assignment of cost $n/10$ when the empty assignment (expected path length of the input tree) has cost $n$ and the optimal assignment has cost 1.

**Statement of the Results.** Our first contribution is the first FPTAS for the 1-HAP. In order to obtain this result, we first prove that for any tree $T$ with $n$ nodes and for any weight function $w$, there is an optimal assignment $A^*$ for $(T, w)$ such that the height of $T^{A^*}$ is at most $O(\log w(T) + \log n)$. Once this result is proved, a pseudo-polynomial time algorithm for the 1-HAP can be obtained by executing the algorithm of [24], mentioned in the previous section, with $D = c(\log w(T) + \log n)$, for a suitable constant $c$. Then, we scale the weights $w$ in a reasonably standard way to obtain the FPTAS. The difficult part in obtaining our FPTAS is proving the bound on the height of $T^{A^*}$– it requires the combination of different kinds of tree decompositions with a non trivial transformation in the optimal tree. These results are presented in Section 3.

Our second contribution is the first constant approximation algorithm for the $k$-HAP. This algorithm recursively decomposes the tree into heavy subtrees of maximum degree $k$ and it can be implemented in $O(n \log n)$ time. It is worth mentioning that our algorithm coincides with the one proposed by Douieb and Langerman [7] for the particular case where $k = 1$. Thus, our analysis here shows that their algorithm provides a constant approximation for the 1-HAP (this was not known before). Although other algorithms with constant approximation do exist for the 1-HAP, the one by Douieb and Langerman has the following advantages: it can be implemented in linear time and it can be dynamized to handle insertions and deletion in logarithmic time. The key idea to obtain our result is a novel lower bound on the cost of the optimal assignment which is much stronger than the entropy-based one given in [3] – roughly speaking, our lower bound is given by a sum of entropy-like functions associated with the trees obtained due to our decomposition. This material is presented in Section 4.

Our third contribution is a linear time algorithm that provides the first constant approximation algorithm for the problem of binary searching in trees when the goal is to minimize the expected number of queries to find a node. This problem was first considered in [18] and it generalizes the classical problem of binary searching in lists with different access probabilities [14, 6]. The current best result for this problem is an $O(\log n)$ approximation [15]. The version of this problem where the goal is to minimize the number of queries in the worst case has been recently addressed [2, 21, 20]. Our motivation for including this result relies on the 'feeling' that the problem of binary searching in trees and the Hotlink Assignment Problem have the same flavor. In fact, the technique employed to analyze the constant approximation algorithm for the k-HAP can be adapted, with some additional effort, to the binary searching problem. This result, together with a formal definition of this search problem, are presented in Section 5.

We shall notice that the complexity of both the 1-HAP and the problem of binary searching in trees remains open. However, our linear time algorithm is interesting for the latter problem even if it is polynomially solvable because it is possible to prove (simple reduction from sorting) that any optimal algorithm for it runs in $\Omega(n \log n)$ time. In addition, it is worth mentioning that the complexity of our problems contrasts with their 'worst case' versions that are polynomially solvable [23, 2, 21]. In fact, the 'average case' versions, studied here, are examples of problems related to searching and coding whose complexities are unknown. Another interesting example is the Huffman coding problem with unequal cost letters [13, 11].

## 2 Preliminaries

As stated in the introduction, given a tree $T$ rooted at node $r$, an assignment $A$ and a weight function $w$, the cost of $A$ under the weights $w$ is given by $\mathrm{EP}(T, A, w) = \sum_{u \in T} d(r, u, T + A) w(u)$. (We omit the weight function when it is clearly understood from the context.) Furthermore, we extend this definition to subtrees of $T$: for any subtree $T'$ of $T$, $\text{EP}(T, A)_{T'} = \sum_{u \in T'} d(r, u, T+A)w(u)$ indicates the expected cost of reaching nodes in $T'$. Also, $\text{OPT}_k(T, w)$ is the cost of the optimal $k$-assignment for $T$ under the weights $w$ (henceforth we use $\text{OPT}(T, w)$ as a shorthand for $\text{OPT}_1(T, w)$). In addition, for any subset $U$ of nodes of $T$, $w(U)$ denotes the sum of the weights of the elements of $U$, namely $w(U) = \sum_{u \in U} w(u)$. For each node $u$ of $T$ we define $T_u$ as the subtree of $T$ composed by all descendants of $u$. We use $r(T')$ to denote the root of a subtree $T'$ of $T$.

A concept that is helpful during the analysis of the results is that of a *non-crossing* assignment. Two hotlinks $(u, a)$ and $(v, b)$ for $T$ are crossing if $u$ is an ancestor of $v$, $v$ is an ancestor of $a$ and $a$ is an ancestor of $b$ (Figure 1.b). An assignment is said to be *non-crossing* if it does not contain crossing hotlinks. Using the definition of the greedy model, it is not difficult to see that any crossing assignment can be transformed into a non-crossing one via removal of some hotlinks, and that these removals do not affect the expected path length.

Now we state two important structural lemmas that allow us to perform transformations on hotlink assignments without increasing much the expected user path length (proof in the appendix).

**Lemma 1** (Multiple Removal Lemma)**.** *Consider a tree $T$ and an assignment $A$ for $T$, where $A$ has $g$ hotlinks leaving $r$ and at most one everywhere else. Then, there is an assignment $A'$ with at most one hotlink per node such that $EP(T, A') \leq EP(T, A) + (g-1)w(T)$.*

**Lemma 2.** *Consider the tree $T$ and let $T'$ be a subtree of $T$. If $v \in T$ is an ancestor of $r(T')$, then $\sum_{u \in T'} d(v, u, T+A)w(u) \geq OPT_g(T')$ for any $g$-assignment $A$.*

The following lemma generalizes the well known fact that every tree $U$ has a node, say $u$, such that all trees in the forest $U \setminus u$ have at most $|U|/2$ nodes.

**Lemma 3.** *Let $U$ be a tree and consider a constant $\alpha$. Then, there is a partition of $U$ into subtrees such that each of these subtrees, except possibly the one containing $r(U)$, has weight not smaller than $\alpha$. In addition, for every tree $U^i$ in the partition, each of the subtrees rooted at children of $r(U^i)$ have weight smaller than $\alpha$.*

## 3 An FPTAS for the 1-Hotlink Assignment Problem

Consider a tree $T$ with $n$ nodes and a weight function $w$. In [24], it is presented a dynamic programming based algorithm for the hotlink assignment problem that has the following property: for each integer $D$, it calculates in $O(n2^D)$ the best 1-assignment among the 1-assignments $A$ such that the height of $T^A$ is at most $D$.

In order to achieve a pseudo-polynomial algorithm and then an FPTAS, we have to argue that for each tree $T$ and weight function $w$ there is an optimal assignment $A^*$ such that the height of $T^{A^*}$ is small, more specifically $O(\log w(T) + \log n)$.

The main idea is to prove that if we walk $c$ steps down in $T^{A^*}$, we reach trees with (geometrically) reduced weight. This result is summarized in the following theorem whose proof is sketched in the end of this section.

**Theorem 1.** *Consider some tree $T$ rooted at node $r$ and an optimal assignment $A^*$ for $T$. Then, there is a constant $c > 2$ (independent of $T$ and $w$) such that for every node $u$ of $T$ with $d(r, u, T + A^*) = c$ we have $w(T_u^{A^*}) \leq \frac{(c-1)w(T)}{c}$.*

It can be shown that $A^*$ must contain an optimal assignment for each subtree $T_u^{A^*}$. Thus, we can use the previous theorem for subtrees of $T^{A^*}$ to argue that every time we walk down $c$ steps, the weight of the subtrees are reduced by at least a constant factor of $(c-1)/c$. As a consequence, it follows that in $O(\log w(T))$ steps we reach subtrees of $T^{A^*}$ that have zero weight. Therefore, if we set $D = G \cdot (\log w(T) + \log n)$[1], for a suitable constant $G$, the dynamic programming algorithm finds an optimal assignment for the 1-Hotlink Assignment Problem in pseudo-polynomial time, that is, $O(2^{O(\log w(T)+\log n)}) = poly(n \cdot w(T))$ time (a more formal proof is deferred to the appendix).

Now we show how to reduce the weight of the tree $T$ in order to obtain in polynomial time an arbitrarily close approximation for the 1-Hotlink Assignment Problem. The argument is rather standard and is the same one used to obtain the FPTAS for the knapsack problem.

Let $W$ be the weight of the heaviest node of $T$ under $w$, namely $W = \max_{u \in T}\{w(u)\}$. Define $K = \frac{\epsilon \cdot W}{n^2}$ and the weight function $w'$ such that $w'(u) = \lceil w(u)/K \rceil$ for every node $u \in T$. Analogously, let $W' = \max_{u \in T}\{w'(u)\}$; notice that $W' = \lceil W/K \rceil \leq (n^2)/\epsilon + 1$. Thus, $w'(T)$ is less than $nW' \leq (n^3)/\epsilon + n$. As a consequence, the dynamic programming algorithm runs in polynomial on $n$ and $1/\epsilon$ over the instance $(T, w')$. Finally, through standard arguments one can prove that $\text{EP}(T, A, w) \leq K \cdot \text{EP}(T, A, w') \leq (1 + \epsilon) \cdot \text{OPT}(T, w)$, assuring the existence of an FPTAS for the 1-hotlink assignment problem (for the full proof we refer to the appendix).

**Theorem 2.** *There exists an FPTAS for 1-HAP.*

**Proof sketch for Theorem 1.** We define $\mathbf{T}_u$ as the subtree of $T_u$ left after some parts of it have been 'adopted' by proper ancestors of $u$ due to the assignment $A^*$ (Figure 2). In other words, a node $v$ belongs to $\mathbf{T}_u$ iff the user path from the root of $T$ to $v$ in $T + A^*$ contains $u$. In order to prove Theorem 1, we assume, by means of contradiction, that $A^*$ satisfies the following hypothesis:

**Hypothesis 1.** *For the constant $c > 2$, given by Theorem 1, there is a node $h \in T$ such that $d(r, h, T + A^*) = c$ and $w(\mathbf{T}_h) > (c - 1)w(T)/c$.*

Notice that because set of nodes of $T_u^{A^*}$ equals the set of nodes of $\mathbf{T}_u$, the hypothesis uses $w(\mathbf{T}_h)$ instead of $w(T_h^{A^*})$, employed in Theorem 1. In the sequel, we transform $A^*$ into an assignment $A$ such that $\text{EP}(T, A) < \text{EP}(T, A^*)$, which contradicts the optimality of $A^*$. Thus, all optimal assignments must satisfy Theorem 1, proving the desired result. Without loss of generality we assume $A^*$ to be non-crossing.

'

Let $h$ be a node of $T$ satisfying Hypothesis 1 and let $Q = (q_1 \to \ldots \to q_{|Q|} = h)$ be the user path from $r$ to $h$ in $T + A^*$. We note that $Q$ cannot have two consecutive hotlinks in $A^*$, otherwise one could obtain an assignment better than $A^*$ through a fairly simple

[1]the additive $\log n$ is a minor technical detail that we avoid explaining for the sake of a better presentation

transformation (formal proof in the appendix). This property of $Q$ will be useful at the end of our analysis. The key idea behind the construction of $A$ is to bring some subtrees of $\mathbf{T}_h$, with similar weights, closer to the root of $T$ through the addition of hotlinks from nodes in $Q$ to nodes in $T_h$. In this process, paths that reach nodes outside $\mathbf{T}_h$ may be lengthened, but because $\mathbf{T}_h$ has most of the weight of $T$ (Hypothesis 1) the expected path length is shortened.

For each node $q_i \in Q \setminus h$ and for each child $j$ of $q_i$, with $j \neq q_{i+1}$, we define $T_i^j = T_j - T_{q_{i+1}}$ (Figure 3). Now we show the first step of the construction of $A$. We start with $A = \emptyset$ and add to $A$ the hotlinks of $A^*$ that have both starting and ending points in $Q$. Then, for each $T_i^j$ we add to $A$ an optimal hotlink assignment $A_i^j$ for $T_i^j$ (Figure 4). Note that some nodes in $Q$ do not have hotlinks anymore – this will be useful later to adopt subtrees from $T_h$.

Now, we show that the difference between the contribution of $T - T_h$ to $\mathrm{EP}(T, A)$ and to $\mathrm{EP}(T, A^*)$ is at most $w(T)$. Note that the trees $\{T_i^j\}$ define a partition of the leaves in $T - T_h$. Fix a tree $T_i^j$. In order to reach a node $u \in T_i^j$ in $T + A$, users needs to reach the root $j$ of $T_i^j$, spending at most $c$ hops, and follow the path from $j$ to $u$ in $T_i^j + A_i^j$. By adding this cost over all nodes in $T_i^j$ we conclude that the contribution of $T_i^j$ to $\mathrm{EP}(T, A)$ is at most $c \cdot w(T_i^j) + \mathrm{OPT}(T_i^j)$. On the other hand, Lemma 2 assures that the contribution of $T_i^j$ to $\mathrm{EP}(T, A^*)$ is not smaller than $\mathrm{OPT}(T_i^j)$. Therefore, the total increase in the cost of reaching nodes of $T - T_h$ is at most $c \cdot w(\bigcup_{i,j} T_i^j) = c \cdot w(T - T_h)$, which from Hypothesis 1 is at most $w(T)$. Although we have not completed the construction of assignment $A$, the next steps will not alter the contribution of $T - T_h$ to $\mathrm{EP}(T, A)$.

Now, we define the second step of the construction of $A$. This step is responsible for bringing nodes of $\mathbf{T}_h$ closer to the root of $T$. Let $D$ be the nodes of $Q$ that are not starting points of hotlinks in $A$ and define $d_i$ as the $i$-th node of $D$ (the $i$th closest to the root of $T$). Let $\{H_1, \ldots, H_k\}$ be the partition of $T_h$ obtained by applying Lemma 3 with $U = T_h$ and $\alpha = (w(T) + 1)/|D|$. We can assume that the trees of our partition are labeled such that for all $i < j$, $r(H_i)$ is not an ancestor of $r(H_j)$ in $T_h$. The second step of the construction consists of adding the hotlinks $\bigcup_{i=1}^k (d_i, r(H_i))$ to $A$. (Figure 5).

The third and last step of our construction is designed to control the expansion of paths that reach nodes of $T_h - \mathbf{T}_h$ in $T + A$. We define $S$ as the set of nodes in $T_h \setminus h$ that are endpoints of hotlinks in $A^*$ that depart from nodes in $Q$. Note that $T_h - \mathbf{T}_h = \{\mathbf{T}_s\}_{s \in S}$. For each meaningful pair $i, j$, we use $H_i^j$ to denote the subtree rooted at the child $j$ of $H_i$. A key property for our construction states that for every $s \in S$, the tree $\mathbf{T}_s$ is a subtree of some $H_i^j$ (proof in the appendix). Define $\overline{H}_i^j$ as the tree obtained by removing from $H_i^j$ its subtrees rooted at nodes in $S$, that is, $\overline{H}_i^j = H_i^j - (\bigcup_{s \in S \cap H_i^j} \mathbf{T}_s)$ (Figure 6.a).

The third step consists of: (i) adding to $A$ an optimal assignment for each of the trees $\{\mathbf{T}_s\}_{s \in S}$ and $\{\overline{H}_i^j\}$; (ii) for each meaningful pair $i, j$, we add hotlinks from $r(H_i^j)$ to all nodes of $S$ that belong to $H_i^j$ (Figure 6.b). Notice that there may exist more than one hotlink leaving the root of some tree $\{H_i^j\}$ – we handle with this situation later. We remark that this transformation does not change the paths reaching nodes outside $T_h$.

In order to understand the gain of this new assignment $A$ we compare the contribution of the nodes of $T_h$ to $\mathrm{EP}(T, A)$ with that to $\mathrm{EP}(T, A^*)$. Since $T_h$ can be partitioned in $\mathbf{T}_h$ and $\{\mathbf{T}_s\}_{s \in S}$ we can analyze the contribution of each of these trees separately.

The trees $\{\mathbf{T}_s\}_{s \in S}$ may provide a loss of $O(w(T))$. In fact, consider some $s \in S$; by

Lemma 2, the contribution of nodes in $\mathbf{T}_s$ to $\mathrm{EP}(T, A^*)$ is at least $\mathrm{OPT}(\mathbf{T}_s)$. On the other hand, in order to reach the nodes of $\mathbf{T}_s$ in $T + A$, users need to reach node $d_i$, spending at most $c$ hops, follow the path $(d_i \rightsquigarrow s)$ in three hops and then walk optimally in $\mathbf{T}_s$. Thus, we can upper bound this cost by $(c + 3) \cdot w(\mathbf{T}_s) + \mathrm{OPT}(\mathbf{T}_s)$. Hence, the loss provided by the trees $\{\mathbf{T}_s\}_{s \in S}$ can be upper bounded by $(c+3) \cdot (\sum_{s \in S} w(\mathbf{T}_s))$, which from Hypothesis 1 is $O(w(T))$.

Now, we argue that $\mathbf{T}_h$ provides a gain of approximately $c \cdot w(\mathbf{T}_h)/2 - O(w(T))$. First, we note that $\mathbf{T}_h = \{\overline{H}_i^j\} \cup \{r(H_i)\}$. To reach the nodes in $\overline{H}_i^j$ in $T + A$ users need to reach the node $d_i$, follow the path $(d_i \to r(H_i) \to j)$ and then walk optimally in $\overline{H}_i^j$. It follows that the contribution of these nodes to $\mathrm{EP}(T, A)$ is at most $d(r, d_i, T + A)w(\overline{H}_i^j) + 2 \cdot w(\overline{H}_i^j) + \mathrm{OPT}(\overline{H}_i^j)$. Similar arguments show that the contribution of $r(H_i)$ to $\mathrm{EP}(T, A)$ is $d(r, d_i, T + A)w(r(H_i)) + w(r(H_i)))$. Thus, by adding the contributions from all trees $\{\overline{H}_i^j\}$ and all roots $\{r(H_i)\}$ we conclude that $\mathrm{EP}(T, A)_{\mathbf{T}_h}$ is at most

$$\sum_i d(r, d_i, T{+}A)w(H_i){+}2w(\mathbf{T}_h){+}\sum_{i,j} \mathrm{OPT}(\overline{H}_i^j) \leq \sum_i d(r, d_i, T{+}A)w(H_i){+}2w(\mathbf{T}_h){+}\mathrm{OPT}(\mathbf{T}_h),$$

where the last inequality follow from multiple applications of Lemma 2. On the other hand, it is clear that $\mathbf{T}_h$ contributes with $c \cdot w(\mathbf{T}_h) + \mathrm{OPT}(\mathbf{T}_h)$ to $\mathrm{EP}(T, A^*)$. Hence,

$$\mathrm{EP}(T, A^*)_{\mathbf{T}_h} - \mathrm{EP}(T, A)_{\mathbf{T}_h} \geq c \cdot w(\mathbf{T}_h) - \sum_i d(r, d_i, T + A)w(H_i) - 2w(\mathbf{T}_h).$$

The gain of $c \cdot w(\mathbf{T}_h)/2 - O(w(T))$, claimed at the beginning of this paragraph, follows from the fact that $\sum_i d(r, d_i, T + A)w(H_i)$ is approximately $c \cdot w(\mathbf{T}_h)/2$. This holds because: (i) the weights of the trees $H_i$ are roughly alike (consequence of Lemma 3); (ii) $\sum_i w(H_i) = w(T_h)$ is just slightly bigger than $w(\mathbf{T}_h)$ (consequence of Hypothesis 1) and (iii) nodes of $D$ can be reached in roughly $c/2$ hops on average. This last fact is a simple consequence of both the definition of $D$ and the property that $Q$ does not contain consecutive hotlinks in $A^*$.

By accounting the contribution of both $T_h = \mathbf{T}_h \cup \{\mathbf{T}_s\}_{s \in S}$ and $T - T_h$ to $\mathrm{EP}(T, A^*)$ and then to $\mathrm{EP}(T, A)$, we conclude that $A$ provides a gain of approximately $(c/2) \cdot w(\mathbf{T}_h) - O(w(T))$.

However, $A$ may not be an 1-assignment since the roots of some trees $\{H_i^j\}$ (and only them) can have more than one hotlink in $A$. We can apply Lemma 1 to replace the assignment of each tree $\{H_i^j\}$ by one with at most one hotlink per node, increasing the cost of the final assignment by at most $\sum_{i,j} |S \cap H_i^j| w(H_i^j)$. Because the weight of $H_i^j$ is not greater than $(w(T) + 1)/|D|$ (definition of partition $\{H_i\}$) and because $|D| \geq |S|$, it follows that this increase is at most $w(T)$. Consequently, the total reduction on the cost from $A^*$ to this new valid assignment is at least $(c/2) \cdot w(\mathbf{T}_h) - O(w(T))$. Since the weight of $\mathbf{T}_h$ is at least $(c - 1)w(T)/c$, the reduction can be made positive by choosing a sufficiently large constant $c$, which implies that the new assignment is better than the optimal one $A^*$, raising a contradiction and completing the proof.

## 4 An O(1) approximation for the $k$-Hotlink Assignment Problem

The algorithm presented in this section is motivated by the observation that entropy provides a good lower bound on the cost of optimal k-assignments for trees of 'low' degree (maximum degree $\approx k$), and that there is an algorithm that works well for these bounded degree trees [8]. Based on these facts, our algorithm greedily decomposes $T$ into subtrees of low degree and then it determines a good assignment for these subtrees using the algorithm of [8]. Since we manage to devise a lower bound on $\mathrm{OPT}_k(T)$ that is given by the sum of the entropies associated with the subtrees obtained through our decomposition, we obtain a constant approximation. We shall note that the same approach is used in the next section for the problem of binary searching in trees.

Consider an input tree $T$ rooted at node $r$. For every node $u \in T$, we define the *cumulative weight* of $u$ as the sum of the weights of its descendants, namely $w(T_u)$. A *heavy k-tree* $Q$ of $T$ is defined recursively as follows: $r$ belong to $Q$; for every node $u$ in $Q$, the $k$ non-leaf children of $u$ with greatest cumulative weight also belong to $Q$ (if there are less than $k$ non-leaf children then all of them belong to $Q$).

For $q \in Q$, if $j$ is a child of $q$ that does not belong to $Q$, then we define $T_q^j$ as the subtree $T_j$. Also define $\overline{T}_q$ as the union of $T_q^j$ over all $j$'s (Figure 7.a). In order to simplify the notation during the analysis, we often omit the range of variables indexing the trees $T_q^j$'s.

The algorithm proceeds as follows: (i) It finds a $k-$heavy tree $Q$ for $T$ and for each $q \in Q$ it defines $w'(q) = w(\overline{T}_q)$; (ii) Calculate a non-crossing $k$-assignment $A_Q$ for the input $(Q, w')$, using the approximation algorithm proposed in [8]; (iii) Calculate recursively a $k$-assignment $A_q^j$ for each input $(T_q^j, w)$ (iv) Output the assignment $A = A_Q \cup \bigcup_{q,j} A_q^j$.

It should be clear from the definition of the algorithm that it outputs a $k$-assignment for $T$. The recursive structure of the algorithm provides the following inequality:

$$\mathrm{EP}(T, A) \leq w(T) + \sum_{q \in Q} d(r, q, Q + A_Q) w(\overline{T}_q) + \sum_{q \in Q} \sum_{j} \mathrm{EP}(T_q^j, A_q^j) \tag{1}$$

For any multiset $W$ we define the entropy of $W$ as $H(W) = -\sum_{w \in W} w \log \frac{w}{\sum_{w' \in W} w'}$. The second term of the righthand side of the above inequality is exactly the cost of the assignment constructed at Step (ii) for the input $(Q, w')$. The analysis of [8] shows that this term is at most $2H(\{w(\overline{T}_q)\})/\log(k+1)$. Therefore, substituting this bound on inequality (1), we have:

$$\mathrm{EP}(T, A) \leq w(T) + 2H(\{w(\overline{T}_q)\})/\log(k+1) + \sum_{q \in Q} \sum_{j} \mathrm{EP}(T_q^j, A_q^j) \tag{2}$$

**Lower bound.** Consider a non-crossing optimal $k$-assignment $A^*$ for $T$. We say that a node $q \in Q$ *captures* a forest $\overline{T}_u$ if it satisfies the following conditions simultaneously: (i) $q$ has either a hotlink or an arc pointing to a node in $\overline{T}_u$; (ii) no proper ancestor of $q$ in $Q$ satisfies (i). Then, we use $c_u$ to denote the node of $Q$ that captures the forest $\overline{T}_u$ (Figure 7.b).

A crucial observation is that every user path in $T + A^*$ that goes to nodes in $\overline{T}_q$ must contain $c_q$ – otherwise, users would have to use a hotlink to 'jump' over $c_q$ and there would be a crossing between this hotlink and the one from $c_q$ pointing to $\overline{T}_q$, which contradicts the fact that $A^*$ is non-crossing (formal proof in the appendix). As a consequence, for every $q \in Q$, the cost of reaching all nodes of $T_q^j$ is:

$$\mathrm{EP}(T, A^*)_{T_q^j} = d(r, c_q, T + A^*)w(T_q^j) + \sum_{u \in T_q^j} d(c_q, u, T + A^*)w(u)$$

Using Lemma 2 with $T' = T_q^j$ and $v = c_q$ to lower bound the last term of the expression, we have $\mathrm{EP}(T, A^*)_{T_q^j} \geq d(r, c_q, T + A^*)w(T_q^j) + \mathrm{OPT}_k(T_q^j)$.

Because we have not included any leaf in $Q$, each leaf of $T$ is in some subtree $\{T_q^j\}$. Therefore,

$$\mathrm{OPT}_k(T) = \sum_{q \in Q} \sum_j \mathrm{EP}(T, A^*)_{T_q^j} \geq \sum_{q \in Q} d(r, c_q, T + A^*)w(\overline{T}_q) + \sum_{q \in Q} \sum_j \mathrm{OPT}_k(T_q^j) \quad (3)$$

Now we use the fact that the first summation of the right hand side of inequality (3) can be seen as the weighted path length of the tree $Q$ where the weight of a node $u \in Q$ is given by the sum of the weights of all forests $\overline{T}_q$ captured by $u$. This observation together with the fact that every node of $Q$ can capture at most $k + 1$ forests, allow us to use Shannon's coding theorem [9] to prove that the first summation of the right hand side of inequality (3) is lower bounded by $\frac{H(\{w(\overline{T}_q)\})}{2 \cdot \log(k+1)} - w(T)$ (proof in the appendix). Combining this bound with inequality (3) we have

$$\mathrm{OPT}_k(T) \geq H(\{w(\overline{T}_q)\})/2\log(k+1) - w(T) + \sum_{q \in Q} \sum_j \mathrm{OPT}_k(T_q^j) \quad (4)$$

When the value of the entropy $H(\{w(\overline{T}_q)\})$ is large enough, it dominates the term $w(T)$ in inequality (4) and leads to a sufficiently strong lower bound. However, when this entropy assumes a small value we need to adopt a different strategy to devise an effective bound. Using properties of the heavy $k$-tree, we can strengthen inequality (3) to obtain the following lower bound, whose proof can be found in the appendix.

$$\mathrm{OPT}_k(T) \geq w(T)/2 + \sum_{q,j} \mathrm{OPT}_k(T_q^j) \quad (5)$$

**Approximation guarantee.** The proof goes by induction on the number of nodes of the tree. From the inductive hypothesis $\mathrm{EP}(T_q^j, A_i^j) \leq \alpha \mathrm{OPT}_k(T_q^j)$, for some constant $\alpha$. When $H(\{w(\overline{T}_q)\})/\log(k + 1) > 3w(T)$, we can compare the upper bound of (2) with the lower bound provided by inequality (4). In the other case, we compare the same upper bound with the lower bound from (5). Straightforward manipulations shows that the inductive step holds for $\alpha \geq 16$ (we refer to the appendix for a full proof). Since the algorithm employed in Step (ii) runs in $O(n \log n)$ time, it is not difficult to argue that our algorithm can be implemented in $O(n \log n)$.

**Theorem 3.** *There exists an $O(n \log n)$ time algorithm that provides constant factor approximation for the k-Hotlink Assignment Problem.*

## 5 Approximation for binary search on trees

Given a tree $T$, we want to find a node $x \in T$ by querying its arcs. (Henceforth when referring to an arc $(i, j)$ the node $i$ is the father of $j$ in $T$.) A query at arc $(i, j)$ has two possible answers: $x$ belongs to the tree $T_j$ or $x$ does not belong to $T_j$. An instance of this problem is a tree $T$ and a weight function $w$ on the nodes of $T$ that indicates the frequency of searching each node. The goal is to find a query strategy that minimizes the weighted average (with respect to $w$) number of queries needed to find nodes of $T$. Every query strategy can be represented by a decision tree $D$ as follows: each internal node of $D$ corresponds to a query for an arc of $T$ and each leaf of $D$ corresponds to a node of $T$. For any decision tree $D$, we use $u_{(i,j)}$ to denote the node of $D$ which corresponds to a query for the arc $(i, j)$ of $T$. In addition, the leaf $u_v \in D$ corresponds to the node $v$ of $T$. Furthermore, each node $u_{(i,j)}$ has exactly two children satisfying the following property: if $u_v$ is the right child of $u_{(i,j)}$ then $v \in T_j$, if $u_v$ is the left child of $u_{(i,j)}$ then $v \in T - T_j - (i, j)$ (Figure 8).

Given a decision tree $D$, the number of queries needed to find a node $v \in T$ is exactly the distance, in number of arcs, from the root of $D$ to $u_v$. Let $d(u, v, D)$ be the distance between nodes $u$ and $v$ in $D$. (When the decision tree is clear from the context we omit the last parameter of this function.) We say that a decision tree $D$ is valid for $T$ if it corresponds to a query strategy for $T$. Therefore, the cost of a valid decision tree is

$$\text{cost}(D, w) = \sum_{v \in Nodes(T)} d(r(D), u_v, D) w(v) \tag{6}$$

Let $Q = (q_1 \to \ldots \to q_{|Q|})$ be a heavy 1-tree (path) of $T$ as defined in the previous section. We define $\overline{T}_{q_i} = T_{q_i} - T_{q_{i+1}}$, for $i < |Q|$ and $\overline{T}_{q_{|Q|}} = T_{q_{|Q|}}$. Note that this last definition is slightly different from the one presented in the previous section since $\overline{T}_{q_i}$ includes $q_i$. Also, we define $T^j_{q_i}$ as the $j$th heaviest maximal subtree rooted at a child of $q_i$ different from $q_{i+1}$ (Figure 9).

Our algorithm has five steps. The first three steps resemble the algorithm of the previous section: (i) find a heavy 1-tree $Q$ of $T$ and for every $q_i \in Q$ define $w'(q_i) = w(\overline{T}_{q_i})$; (ii) compute a decision tree $D'$ for $(Q, w')$ using the algorithm of [6]; (iii) calculate recursively a decision tree $D^j_i$ for each subtree $T^j_{q_i}$.

Let $e^j_i$ be the arc of $T$ joining $q_i$ to $T^j_{q_i}$ and let $n_i$ be the number of children of $q_i$ different from $q_{i+1}$. The fourth step consists of building a decision tree $D_i$ for each $\overline{T}_{q_i}$ as follows. The leftmost path of $D_i$ consists of the nodes corresponding to the arcs $e^1_i, \ldots, e^{n_i}_i$, with $u_{q_i}$ appended at the end. In addition, for every $j$, $D^j_i$ is the right child of the node corresponding to $e^j_i$ in $D_i$ (Figure 10). (Notice this constitutes a strategy that starts testing arcs from the heavier trees $\{T^j_i\}$ first.) Finally, in the fifth step, for every $q_i \in Q$, we replace the leaf $u_{q_i}$ in $D'$ by $D_i$. (Figure 11)

It is not difficult to verify that the construction of the algorithm implies that

$$\text{cost}(D, w) = \text{cost}(D', w') + \sum_{i,j} j \cdot w(T^j_{q_i}) + \sum_{i,j} \text{cost}(D^j_i, w) + \sum_{q_i} n_i w(q_i) \tag{7}$$

**Lower bound.** Let $D^*$ be a minimum cost decision tree for $(T, w)$, rooted at $r^*$. In addition, let $\text{OPT}(T, w)$ be its cost. The following property is crucial for our analysis:

every subtree $T'$ of $T$ contains a *representative* in $D^*$, that is, a node in $D^*$ that corresponds to either an arc or node of $T'$ and it is an ancestor in $D^*$ of all other nodes corresponding to arcs and nodes of $T'$. We use $u(T')$ to denote the *representative* of $T'$ in $D^*$

Consider a node $v \in T^j_{q_i}$. Employing the above property with $T' = T^j_{q_i}$ and then with $T' = \overline{T}_{q_i}$, we have that the path from $r^*$ to $u_v$ is in fact $(r^* \rightsquigarrow u(\overline{T}_{q_i}) \rightsquigarrow u(T^j_{q_i}) \rightsquigarrow u_v)$. Now consider a node $q_i \in Q$; again using the above property, the path from $r^*$ to $u_{q_i}$ is $(r^* \rightsquigarrow u(\overline{T}_{q_i}) \rightsquigarrow u_{q_i})$. Because all nodes of $T$ are in $\{T^j_{q_i}\}$ or in $Q$, we can write the cost of $D^*$ as:

$$\begin{aligned}\mathrm{cost}(D^*, w) = \mathrm{OPT}(T, w) \quad &= \sum_{q_i} d(r^*, u(\overline{T}_{q_i}))w(\overline{T}_{q_i}) + \sum_{q_i,j} d(u(\overline{T}_{q_i}), u(T^j_{q_i}))w(T^j_{q_i}) \\ &+ \sum_{q_i,j} \sum_{v \in Nodes(T^j_{q_i})} d(u(T^j_{q_i}), u_v)w(v) + \sum_{q_i} d(u(\overline{T}_{q_i}), u_{q_i})w(q_i) \quad (8)\end{aligned}$$

The first term of (8) can be seen as the cost of $D^*$ under a cost function where the representative of $\overline{T}_{q_i}$, for every $i$, has weight $w(\overline{T}_{q_i})$ and all other nodes have weight 0. Since $D^*$ is a binary tree, we can use Shannon coding theorem to guarantee that the first term of (8) is lower bounded by $H(\{w(\overline{T}_{q_i})\})/c - w(T)$, for a suitable constant $c$.

Now we bound the second term of (8). Fix $q_i$; we can prove (appendix) that for any level $\ell$ of $D^*$ there are at most two representatives of the trees $T^j_{q_i}$ located at $\ell$. This together with the fact that $T^j_{q_i}$ is the $j$th heaviest tree rooted at a child of $q_i$ guarantee that

$$\sum_j \lfloor (j-1)/2 \rfloor w(T^j_{q_i}) \le \sum_j d(u(\overline{T}_{q_i}), u(T^j_{q_i}))w(T^j_{q_i})$$

Thus, the second term of (8) can be lower bounded by $\sum_{q_i,j} \lfloor (j-1)/2 \rfloor w(T^j_{q_i})$

For the third term of (8) we fix a tree $T^j_{q_i}$ and then we construct a tree $S$ associated with $T^j_{q_i}$ as follows: the nodes of $S$ correspond to the arcs and nodes of $T^j_{q_i}$; there is an arc from $u$ to $v$ in $S$ iff $u$ is the closest ancestor of $v$ in $D^*$, among the nodes of $S$ (Figure 12). By construction the distance between two nodes $u$ and $v$ in $S$ is not larger than that distance in $D^*$. Thus,

$$cost(S, w) = \sum_{v \in Nodes(T^j_{q_i})} d(u(T^j_{q_i}), u_v, S)w(v) \le \sum_{v \in Nodes(T^j_{q_i})} d(u(T^j_{q_i}), u_v, D^*)w(v)$$

In addition, it possible to prove that $S$ is a valid decision tree for $T^j_{q_i}$ which implies that $\mathrm{OPT}(T^j_{q_i}, w) \le cost(S, w)$. It follows that the third term of (8) is at least $\sum_{q_i,j} \mathrm{OPT}(T^j_{q_i}, w)$.

Finally, for the fourth term we fix $q_i$ and note that the path in $D^*$ connecting $u(\overline{T}_{q_i})$ to $u_{q_i}$ must contain the nodes corresponding to arcs $e^1_i, \ldots, e^{n_i}_i$. Applying this reasoning for each $q_i$, we conclude that last term of (8) is lower bounded by $\sum_{q_i} n_i \cdot w(q_i)$.

Therefore, applying the previous discussion to lower bound the terms of (8) we obtain that

$$\mathrm{OPT}(T, w) \ge \frac{H(\{w(\overline{T}_{q_i})\})}{c} - w(T) + \sum_{q_i,j} \lfloor (j-1)/2 \rfloor w(T^j_{q_i}) + \sum_{q_i,j} \mathrm{OPT}(T^j_{q_i}, w) + \sum_{q_i} n_i w(q_i) \quad (9)$$

As in the previous section, a different lower bound on $\mathrm{OPT}(T, w)$ is required when the value of the entropy is small. Again using properties guaranteed by the construction of the heavy path, we can prove the following bound: $\mathrm{OPT}(T, w) \geq w(T)/2 + \sum_{q_i,j} \mathrm{OPT}(T^j_{q_i}, w)$

**Approximation.** Using the analysis provided in [6], we can upper bound first term of inequality (7) by $H\{w(\overline{T}_{q_i})\} + 2w(T)$. The proof of the constant approximation is similar to the one given in the previous section. It goes by induction on the subtrees of $T$ and uses the different lower bounds depending on the value of the entropy $H(\{w(\overline{T}_{q_i})\})$. For the running time, we note that the decomposition into heavy 1-trees can be done in linear time. Thus, the bottleneck of the algorithm is sorting the subtrees rooted at children of $q_i$ at Step (iv). However, it is possible to implement an approximate sorting (scaling the weight function $w$) in linear time and still guarantee a constant factor approximation.

**Theorem 4.** *There is a linear time algorithm which provides a constant factor approximation for the problem of binary searching in trees.*

**Acknowledgments** We would like to thank Jose Correa and Marcos Kiwi for helpful discussions during this research.

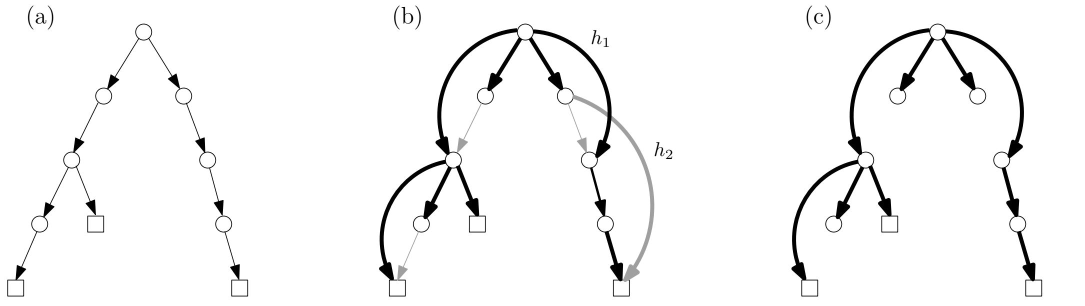

Figure 1: (a) Original tree. (b) Enhanced tree, with greedy paths in bold. Hotlinks $h_1$ and $h_2$ are crossing. (c) Tree induced by user paths.

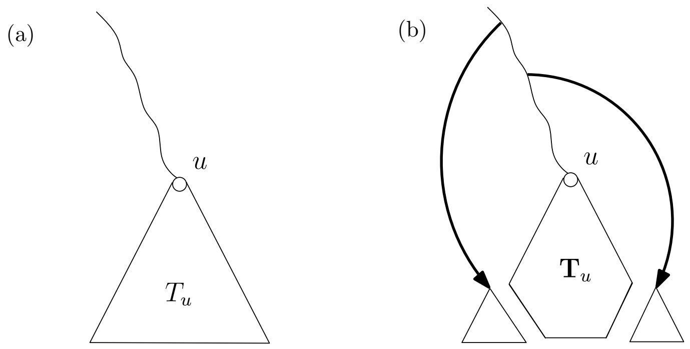

Figure 2: (a) Tree $T_u$. (b) Illustration of tree $\mathbf{T}_u$

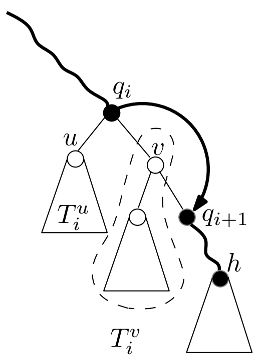

Figure 3: Construction of trees $T_i^j$. The path $Q$ is depictured in bold.

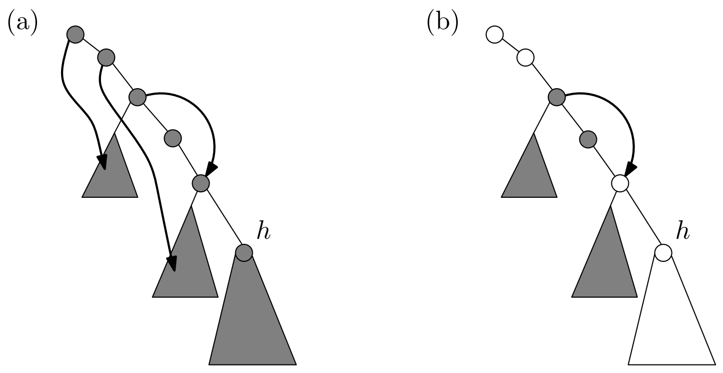

Figure 4: (a) Illustration of assignment $A^*$. (b) Illustration of the first part of the construction of $A$. The shaded nodes and shaded subtrees may contain hotlinks, but not the blank ones.

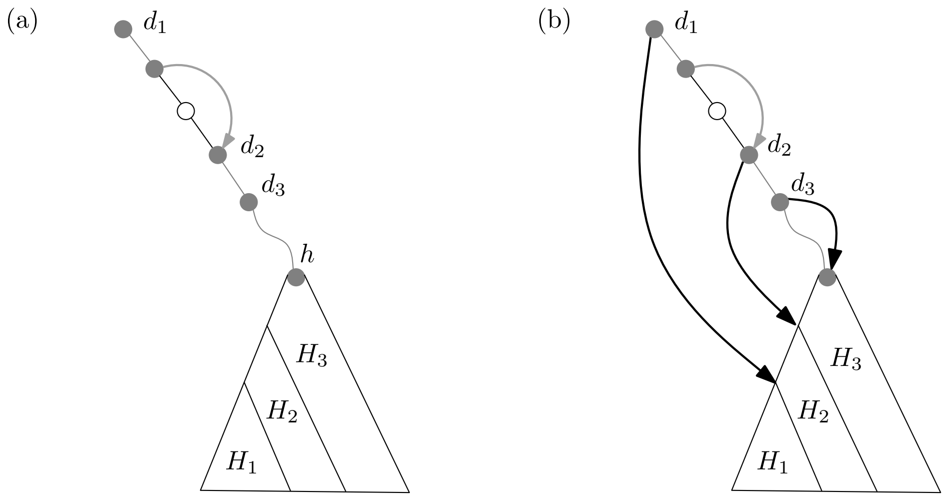

Figure 5: (a) Enhanced tree $T + A$ after the first step of the construction, with the path $Q$ in gray. (b) Addition of hotlinks $(d_i, r(H_i))$, shortening the paths to nodes of $T_h$.

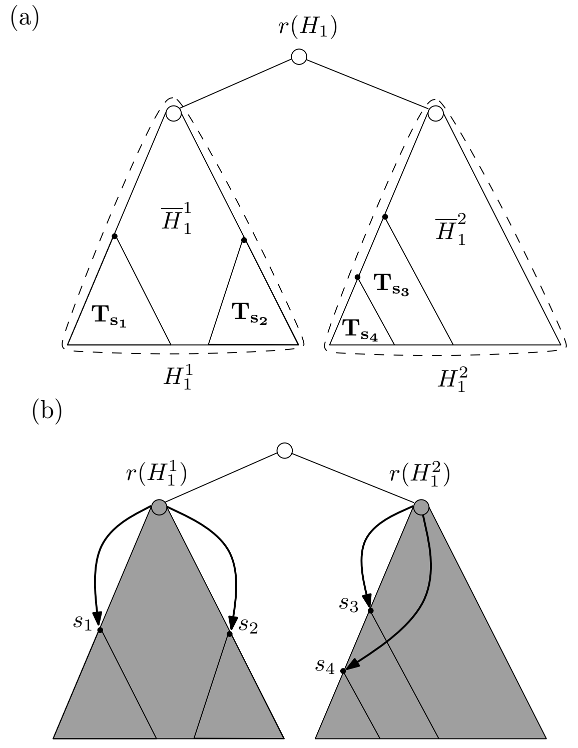

Figure 6: Illustration of the third part of the construction of $A$. (a) Tree $H_1$ and its subtrees $\{H_i^j\}$, $\{\overline{H}_i^j\}$ and $\{\mathbf{T}_s\}_{s\in S}$. (b) Addition of hotlinks $(r(H_1^j), s)$ for $s \in S \cap H_1^j$ and optimal assignments for the subtrees $\overline{H}_1^1$, $\overline{H}_1^2$, $\mathbf{T}_{s_1}$, $\mathbf{T}_{s_2}$, $\mathbf{T}_{s_3}$ and $\mathbf{T}_{s_4}$.

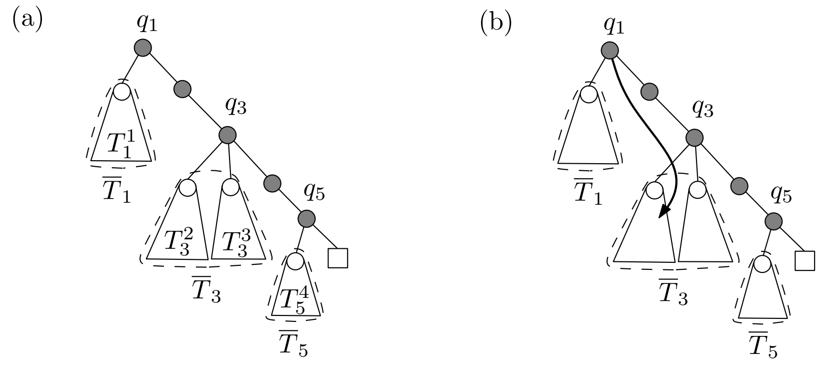

Figure 7: (a) Illustration of structures $T_i^j$ and $\overline{T}_i$, with the nodes of the heavy 1-tree in gray. (b) Node $q_1$ captures $\overline{T}_{q_3}$, hence $c_3 = q_1$. Also, $q_1$ and $q_5$ capture their own forests $\overline{T}_{q_1}$ and $\overline{T}_{q_5}$.

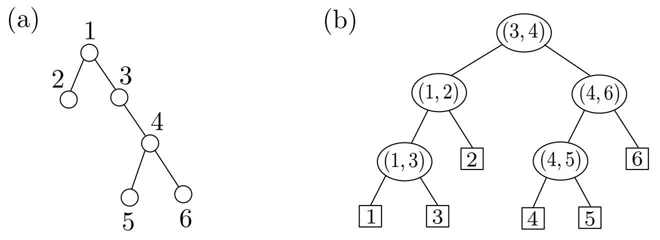

Figure 8: (a) Tree $T$. (b) Example of a decision tree for $T$; Internal nodes correspond to arcs of $T$ and leaves to nodes of $T$

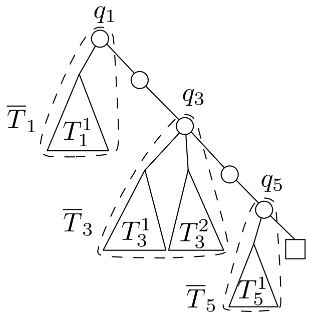

Figure 9: Example of structures $\overline{T}_q$ and $T_q^j$.

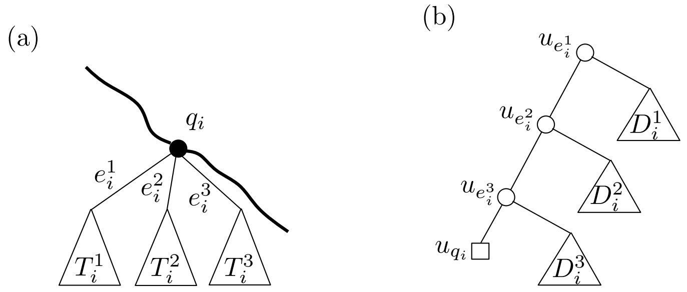

Figure 10: Illustration of Step (iv). (a) Tree $T$ with the heavy path in bold. (b) Decision tree $D_i$ for $\overline{T}_i$.

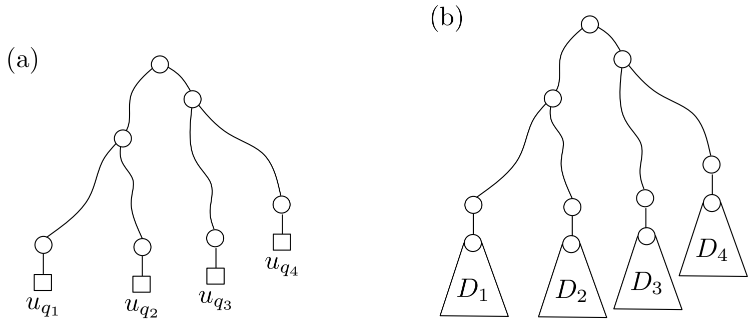

Figure 11: Illustration of Step (v). (a) Decision tree $D'$ built at Step (i). (b) Decision tree $D$ constructed by replacing the nodes $\{u_{q_i}\}$ by the decision trees $\{D_i\}$.

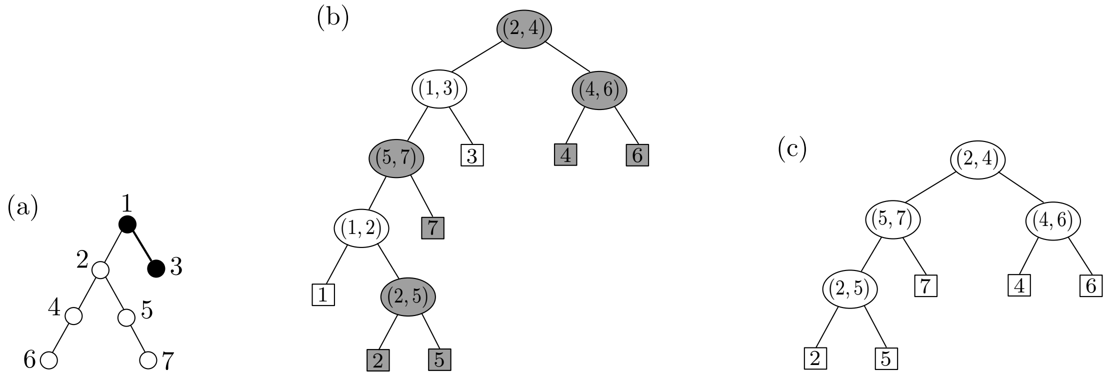

Figure 12: Construction of tree $S$. (a) Tree $T$ with heavy path in black. (b) Decision $D$ tree for $T$, with nodes corresponding to nodes and arcs of $T_1^1 = T_2$ in gray. (c) Decision tree $S$ for $T_1^1$ constructed by connecting the nodes of $D$ corresponding to nodes and arcs of $T_1^1$.

## References

[1] M. Adler and B. Maggs. Protocols for asymmetric communication channels. *JCSS: Journal of Computer and System Sciences*, 63, 2001.

[2] Y. Ben-Asher, E. Farchi, and I. Newman. Optimal search in trees. *SICOMP: SIAM Journal on Computing*, 28, 1999.

[3] P. Bose, E. Kranakis, D. Krizanc, M. Martin, J. Czyzowicz, A. Pelc, and L. Gasieniec. Strategies for hotlink assignments. In *International Symposium on Algorithms and Computation*, pages 23–34, 2000.

[4] P. Bose, D. Krizanc, S. Langerman, and P. Morin. Asymmetric communication protocols via hotlink assignments. *Theory Comput. Syst.*, 36(6):655–661, 2003.

[5] J. Czyzowicz, E. Kranakis, D. Krizanc, A. Pelc, and M. V. Martin. Enhancing hyperlink structure for improving web performance. *Journal of Web Engineering*, 1(2):93–127, March 2003.

[6] R. de Prisco and A. de Santis. On binary search trees. *Inf. Process. Lett.*, 45(5):249–253, 1993.

[7] K. Douïeb and S. Langerman. Dynamic hotlinks. In *Proceedings of the 9th Workshop on Algorithms and Data Structures (WADS 2005)*, volume 3608 of *LNCS*, pages 182–194. Springer-Verlag, 2005.

[8] K. Douïeb and S. Langerman. Near-entropy hotlink assignments. In *Proceedings of the 14th Annual European Symposium on Algorithms (ESA 2006)*, volume 4168 of *LNCS*, pages 292–303, Zürich, Switzerland, 2006. Springer Berlin / Heidelberg.

[9] R. Gallager. *Information Theory and Reliable Communication.* John Wiley & Sons, Inc., New York, NY, USA, 1968.

[10] O. Gerstel, S. Kutten, R. Matichin, and D. Peleg. Hotlink enhancement algorithms for web directories: (extended abstract). In T. Ibaraki, N. Katoh, and H. Ono, editors, *ISAAC*, volume 2906 of *Lecture Notes in Computer Science*, pages 68–77. Springer, 2003.

[11] M. Golin, C. Kenyon, and N. Young. Huffman coding with unequal letter costs. In *STOC: ACM Symposium on Theory of Computing (STOC)*, 2002.

[12] T. Jacobs. Constant factor approximations for the hotlink assignment problem. In F. K. H. A. Dehne, J.-R. Sack, and N. Zeh, editors, *WADS*, volume 4619 of *Lecture Notes in Computer Science*, pages 188–200. Springer, 2007.

[13] R. Karp. Minimum-redundancy coding for the discrete noiseless channel. *IRE Trans. Inform. Theory*, 7:27–39, 1961.

[14] D. Knuth. Optimum binary search trees. *Acta. Informat.*, 1:14–25, 1971.

[15] R. Kosaraju, T. Przytycka, and R. Borgstrom. On an optimal split tree problem. In *WADS: 6th Workshop on Algorithms and Data Structures*, 1999.

[16] E. Kranakis, D. Krizanc, and S. Shende. Approximate hotlink assignment. *Lecture Notes in Computer Science*, 2223:756–767, 2001.

[17] S. Kutten, O. Gerstel, E. Laber, R. Matichin, D. Peleg, A. Pessoa, and C. Souza. Reducing human interactions in web directory searches. *ACM Transactions on Information Systems*, (25), 2007.

[18] M. Lipman and J. Abrahams. Minimum average cost testing for partially ordered components. *IEEE Transactions on Information Theory*, 41(1):287–291, 1995.

[19] R. Matichin and D. Peleg. Approximation algorithm for hotlink assignment in the greedy model. In *International Colloquium on Structural Information and Communication Complexity (SIROCCO), LNCS*, volume 11. 2004.

[20] S. Mozes, K. Onak, and O. Weimann. Finding an optimal tree searching strategy in linear time. In *Proceedings ACM-SIAM Symposium on Discrete Algorithms 2008*. ACM/SIAM, 2008.

[21] K. Onak and P. Parys. Generalization of binary search: Searching in trees and forest-like partial orders. In *FOCS*, pages 379–388, 2006.

[22] M. Perkowitz and O. Etzioni. Adaptive web sites: an AI challenge. In *IJCAI (1)*, pages 16–23, 1997.

[23] A. Pessoa, E. Laber, and C. de Souza. Efficient algorithms for the hotlink assignment problem: The worst case search. In *ISAAC*, pages 778–792, 2004.

[24] A. Pessoa, E. Laber, and C. Souza. Efficient implementation of hotlink assignment algorithms for web sites. In *Proceedings of ALENEX*, 2004.

## Appendix

## A Preliminary lemmas

In order to simplify the notation of the proofs, for any tree $T$ and assignment $A$, we define $A|T$ as the subset of hotlinks of $A$ with both endpoints in $T$. Before starting to present proofs for the preliminary lemmas, we need to define more formally the trees $\mathbf{T}_u(A)$:

**Definition 1.** *Consider a tree $T$ and a non-crossing assignment $A$ for it. Let $u$ be a node in $T$. Then we have the following equivalent definitions:*

*(i) Let $U = \{v \in T :$ user path from $r$ to $v$ in $T + A$ traverses node $u\}$. Then $\mathbf{T}_u(A)$ is the subgraph of $T$ induced by $U$.*

*(ii) Let $U = \{v : (w, v) \in A$ and $v$ is a proper descendant of $u$ and $w$ is a proper ancestor of $u\}$. Then $\mathbf{T}_u(A) = T_u - \left(\bigcup_{v\in U} T_v\right)$.*

The next proposition is a direct implication of th definition of a valid hotlink assignment.

**Proposition 1.** *Consider a tree $T$ and an assignment $A$ for it. Let $u$ and $v$ be nodes in $T$ such that $v \in T_u$. Let $T'$ be a subtree of $T$ that contains both $u$ and $v$. Then, the user path from $u$ to $v$ in $T + A$ equals to the user path from $u$ to $v$ in $T' + A$, and consequently in $T' + A|T'$.*

**Lemma 4** (Local Change Lemma)**.** *Consider some tree $T$ rooted at $r$ and a non-crossing assignment $A$ for it. Let $U$ be a subsets of nodes of $T$ such that the trees $\{\mathbf{T}_u(A)\}_{u\in U}$ are pairwise disjoint. For each $u \in U$ let $A_u$ be the hotlinks of $A$ with both endpoints in $\mathbf{T}_u(A)$ and let $A'_u$ another assignment whose hotlinks have both endpoints in $\mathbf{T}_u(A)$. Finally define $A' = A - (\bigcup_{u\in U} A_u) \cup (\bigcup_{u\in U} A'_u)$. Then the following holds: (i) $\mathbf{T}_u(A') = \mathbf{T}_u(A)$ (ii) if $v$ does not belong to any $\{\mathbf{T}_u(A)\}_{u\in U}$, then $d(r, v, T + A) = d(r, v, T + A')$; (iii) if $u \in U$ and $v \in \mathbf{T}_u(A)$, then $d(r, v, T + A') = d(r, u, T + A) + d(u, v, \mathbf{T}_u(A) + A'_u)$.*

*Proof.* (i) Consider some $u \in U$. Let $(x, y)$ be a hotlink in such that $x$ is a proper ancestor of $u$ and $y$ is a proper descendant of $y$. Clearly both $x$ and $y$ cannot belong to a single subtree of $T - \mathbf{T}_u(A)$. Because the trees $\{\mathbf{T}_u(A)\}_{u\in U}$ are disjoint, this means that $(x, y)$ does not belong to any tree $\{\mathbf{T}_u(A)\}_{u\in U}$ and consequently such hotlink is present in $A'$ iff it is also present in $A$. The result follows by the definition of the trees $\mathbf{T}_u(A)$ and $\mathbf{T}_u(A')$.

(ii) Notice that the path from $r$ to a node $v \notin \bigcup_{u\in U} \mathbf{T}_u(A)$ in $T + A$ cannot contain a node $x \in \bigcup_{u\in U} \mathbf{T}_u(A)$, otherwise the path would be $(r \rightsquigarrow u \rightsquigarrow x)$ for some $u \in U$ plus $(x \rightsquigarrow v)$, implying $v \in \bigcup_{u\in U} \mathbf{T}_u(A)$ (remember that user paths are unique). Then for all nodes in the path $r$ to $v$ in $T + A$ the hotlinks in $A'$ are the same as in $A$. As the users have the same options of arcs and hotnlinks in $T + A$ and $T + A'$ when going from $r$ to a node outside $\bigcup_{u\in U} \mathbf{T}_u(A)$, the path they chose must be the same in $T + A$ and $T + A'$. (Notice that this argument also holds for the paths from $v$ to the nodes of $U$.)

(iii) Consider a node $u \in U$ and let $v$ be a node in $\mathbf{T}_u(A)$. From (i) we have that $d(r, v, T + A') = d(r, u, T + A') + d(u, v, T + A')$. From (ii) we have $d(r, u, T + A') = d(r, u, T + A)$. Because the tree $\mathbf{T}_u(A)$ contains all nodes in the path of $T$ from $u$ to $v$, the definition of a valid hotlink assignment implies that only nodes in $\mathbf{T}_u(A)$ can be used when going from $u$ to $v$. Therefore $d(u, v, T + A') = d(u, v, \mathbf{T}_u(A) + A'_u)$ and the result holds. $\square$

**Corollary 1.** *Consider some tree $T$ rooted at $r$ and a non-crossing assignment $A$ for it. Let $U$ be a subsets of nodes of $T$ such that the trees $\{\mathbf{T}_u(A)\}_{u\in U}$ are pairwise disjoint. For each $u \in U$ let $A_u$ be the hotlinks of $A$ with both endpoints in $\mathbf{T}_u(A)$ and let $A'_u$ another assignment whose hotlinks have both endpoints in $\mathbf{T}_u(A)$. Finally define $A' = A - (\bigcup_{u\in U} A_u) \cup (\bigcup_{u\in U} A'_u)$. Then:*

$$EP(T, A') = EP(T, A) + \sum_{u\in U} (EP(\mathbf{T}_u(A), A') - EP(\mathbf{T}_u(A), A)) \tag{10}$$

*Proof.* From Lemma 4, for all $u \in U$ $\mathbf{T}_u(A') = \mathbf{T}_u(A)$ , and thus the trees $\{\mathbf{T}_u(A')\}$ are pairwise disjoint. Therefore we can write the cost of $A'$ as:

$$\begin{aligned}
\mathrm{EP}(T, A') &= \sum_{u\in U} \mathrm{EP}(T, A')_{\mathbf{T}_u(A')} + \mathrm{EP}(T, A')_{T-\bigcup_{u\in U}\mathbf{T}_u(A')} \\
&= \sum_{u\in U} \big(d(r, u, T + A')w(\mathbf{T}_u(A')) + \mathrm{EP}(\mathbf{T}_u(A'), A')\big) + \mathrm{EP}(T, A')_{T-\bigcup_{u\in U}\mathbf{T}_u(A')} \\
&= \sum_{u\in U} \big(d(r, u, T + A)w(\mathbf{T}_u(A)) + \mathrm{EP}(\mathbf{T}_u(A), A'_u)\big) + \mathrm{EP}(T, A)_{T-\mathbf{T}_u(A)}
\end{aligned}$$

where the third equality follows from properties of Lemma 4.

By similar derivation, we have that:

$$\mathrm{EP}(T, A) = \sum_{u\in U} (d(r, u, T + A)w(\mathbf{T}_u(A)) + \mathrm{EP}(\mathbf{T}_u(A), A_u)) + \mathrm{EP}(T, A)_{T-\mathbf{T}_u(A)}$$

Comparing the expressions for $\mathrm{EP}(T, A)$ and $\mathrm{EP}(T, A')$ it is easy to see that the result holds. $\square$

**Lemma 5** (Multiple pushdown)**.** *Consider a tree $T$ rooted at node $r$ and a $k$-hotlink assignment $A$ for $T$. There is a $k$-hotlink assignment $A'$ for $T$ such that $r$ does not have any hotlink in $A'$ and that $EP(T, A') \leq EP(T, A) + w(T)$.*

*Proof.* Without loss of generality, assume that $A$ is non-crossing and that it does not contain proper hotlinks.

The proof goes by induction on the number of nodes of $T$, with the trivial base case when the tree is just a node. Suppose that the result holds for any tree $T'$ with fewer nodes than $T$. Let $\delta(r)$ be the children of $r$ in $T$, and for each node $i \in \delta(r)$ let $\sigma_i = \{j \in T_i : (r, j) \in A\}$. Because there are only proper hotlinks in $A$, each $j \in \sigma_i$ must be a proper descendant of $i$, and it follows that the trees $\mathbf{T}_i(A)$ and $\{\mathbf{T}_j(A)\}_{j\in\sigma_i}$ form a partition of nodes of $T_i$. For any node $i \in \delta(r)$, the path to reach a node $u$ in $\mathbf{T}_i(A)$ is $(r \to i \rightsquigarrow u)$, and weighting for all $u \in \mathbf{T}_i(A)$ we have $\mathrm{EP}(T, A)_{\mathbf{T}_i(A)} = w(\mathbf{T}_i(A)) + \mathrm{EP}(\mathbf{T}_i(A), A_i)$, where $A_i = A|\mathbf{T}_i(A)$. Also, for any node $j \in \sigma_i$, the path from $r$ to $u \in \mathbf{T}_j(A)$ is $(r \to j \rightsquigarrow u)$. Thus $\mathrm{EP}(T, A)_{\mathbf{T}_j(A)} = w(\mathbf{T}_j(A)) + \mathrm{EP}(\mathbf{T}_j(A), A_j)$, where $A_j = A|\mathbf{T}_j(A)$. Because the weight of the root of $T$ is zero, the total cost of reaching nodes in $T$ is then given by the sum of the cost of reaching nodes in $\{T_i\}_{i\in\delta(r)}$:

$$\begin{aligned}\mathrm{EP}(T,A) &= \sum_{i\in\delta(r)} \mathrm{EP}(T,A)_{T_i} = \sum_{i\in\delta(r)} \mathrm{EP}(T,A)_{\mathbf{T}_i(A)} + \sum_{i\in\delta(r)}\sum_{j\in\sigma_i} \mathrm{EP}(T,A)_{\mathbf{T}_j(A)} \\ &= w(T) + \sum_{i\in\delta(r)} \mathrm{EP}(\mathbf{T}_i(A),A_i) + \sum_{i\in\delta(r)}\sum_{j\in\sigma_i} \mathrm{EP}(\mathbf{T}_j(A),A_j)\end{aligned}$$

By the inductive hypothesis, for each $i\in\delta(r)$ we can find an assignment $A'_i$ for $\mathbf{T}_i(A)$ with no hotlinks in $i$ which satisfies $\mathrm{EP}(\mathbf{T}_i(A),A'_i) \le \mathrm{EP}(\mathbf{T}_i(A),A_i) + w(\mathbf{T}_i(A))$. Then we define the assignment $A' = \bigcup_{i\in\delta(r)}(A'_i \cup \bigcup_{j\in\sigma_i}(A_j \cup (i,j)))$. Notice that because $A'_i$ does not have hotlinks in $i$, there are at most $|\sigma_i|$ hotlinks in $i$ in $A'$, which is less than $k$. Also, from the fact that $\mathbf{T}_i(A)$ and $\{\mathbf{T}_j(A)\}_{j\in\sigma_i}$ are disjoint it follows that there are at most $k$ hotlinks on every other nodes of $T$ in $A'$. Again, the cost of reaching nodes of $\mathbf{T}_i(A)$ for $i\in\delta(r)$ is $w(\mathbf{T}_i(A))+\mathrm{EP}(\mathbf{T}_i(A),A'_j)$. Now the path from $r$ to a node $u\in\mathbf{T}_j(A)$ for $j\in\sigma_i$ is $(r\to i\to j \rightsquigarrow u)$. Thus, $\mathrm{EP}(T,A')_{\mathbf{T}_j(A)} = 2w(\mathbf{T}_j(A)) + \mathrm{EP}(\mathbf{T}_j(A),A_j)$. Consequently:

$$\begin{aligned}\mathrm{EP}(T,A') &= \sum_{i\in\delta(r)} (w(\mathbf{T}_i(A)) + \mathrm{EP}(\mathbf{T}_i(A),A'_i)) + \sum_{i\in\delta(r)}\sum_{j\in\sigma_i}(2w(\mathbf{T}_j(A)) + \mathrm{EP}(\mathbf{T}_j(A),A_j)) \\ &\le \sum_{i\in\delta(r)} (2w(\mathbf{T}_i(A)) + \mathrm{EP}(\mathbf{T}_i(A),A_i)) + \sum_{i\in\delta(r)}\sum_{j\in\sigma_i}(2w(\mathbf{T}_j(A)) + \mathrm{EP}(\mathbf{T}_j(A),A_j)) \\ &= \mathrm{EP}(T,A) + w(T)\end{aligned}$$

□

**Lemma 1** (Multiple Removal Lemma). *Consider a tree $T$ and an assignment $A$ for $T$, where $A$ has $g$ hotlinks leaving $r$ and at most one everywhere else. Then, there is an assignment $A'$ with at most one hotlink per node such that $EP(T,A') \le EP(T,A) + (g-1)w(T)$.*

*Proof.* The proof goes by induction on $g$. Suppose it holds for $g' < g$. Let $v$ be the node further away from $r$ (namely with greatest $d(r,v,T)$) such that $(r,v)\in A$. Let $A_2 = A|(T-T_v)$. Notice $A_2$ has $g-1$ hotlinks in $r$, because $(r,v)$ does not belong to it. It is easy to see that $\mathrm{EP}(T,A) = \mathrm{EP}(T-T_v,A_2) + w(T_v) + \mathrm{EP}(T_v,A_v)$, where $A_v = A|T_v$. By induction find an assignment $A'_2$ for $T-T_v$ with at most one hotlink per node and such that $\mathrm{EP}(T-T_v,A'_2) \le \mathrm{EP}(T-T_v,A_2) + (g-2)\cdot w(T)$. Now apply Lemma 5 to find an assignment $A''_2$ with no hotlink in $r$ such that $\mathrm{EP}(T-T_v,A''_2) \le \mathrm{EP}(T-T_v,A'_2) + w(T) \le \mathrm{EP}(T-T_v,A_2) + (g-1)\cdot w(T)$. Define $A' = A''_2 \cup A_v \cup (r,v)$. Clearly $A'$ has at most one hotlink in $r$. Again we have that $\mathrm{EP}(T,A') = \mathrm{EP}(T-T_v,A''_2) + w(T_v) + \mathrm{EP}(T_v,A_v)$, and the result follows. □

**Lemma 2.** *Consider the tree $T$ and let $T'$ be a subtree of $T$. If $v\in T$ is an ancestor of $r(T')$, then $\sum_{u\in T'} d(v,u,T+A)w(u) \ge OPT_g(T')$ for any $g$-assignment $A$.*

*Proof.* Let $U = \{u_1 \to \ldots \to u_{|U|}\}$ be the path from $v$ to $r'$ in $T$. Define the tree $T^+ = U\cup T'$ and let $A^+ = A|T^+$. Notice that the definition of a valid hotlink assignment implies that only arcs and hotlinks in $T^+ + A^+$ can be used when going from $v$ to nodes in $T'$ in $T+A$. Therefore, $\mathrm{EP}(T,A)_{T'} = \mathrm{EP}(T^+,A^+)$. We sequentially apply Lemma 5 to nodes $u_i$, starting at node $u_1$, then at $u_2$, and so forth. At the end, we have a $g$-assignment

$A'$ for $T'$ such that $\mathrm{EP}(T^+, A') \leq \mathrm{EP}(T, A)_{T'} + |U| \cdot w(T')$. But also notice that because there are no hotlinks in the path from $v$ to $r'$ in $T^+ + A^+$, the path from $v$ to a node $u \in T'$ in $T^+ + A^+$ is $(u_1 \to \ldots u_{|U|} \rightsquigarrow u)$. Therefore, the cost of reaching nodes of $T^+$ (which is exactly the same cost of reaching nodes of $T'$, as only nodes of $T'$ have non-zero weights) is $\mathrm{EP}(T^+, A') = |U| \cdot w(T') + \mathrm{EP}(T', A')$. Because the cost $\mathrm{EP}(T, A)_{T'}$ equals the cost $\mathrm{EP}(T^+, A^+)$, we have $\mathrm{EP}(T, A)_{T'} \geq \mathrm{EP}(T', A')$, which is also not less than $\mathrm{OPT}_g(T')$ and the result follows. $\square$

**Lemma 6.** *Let $U$ be a tree and consider a constant $\alpha$. Then, there is a partition of $U$ into subtrees such that each of these subtrees, except possibly the one containing $r(U)$, has weight not smaller than $\alpha$. In addition, for every tree $U^i$ in the partition, each of the subtrees rooted at children of $r(U^i)$ have weight smaller than $\alpha$.*

*Proof.* The proof goes by induction on the subtrees of $U$. Notice that when $U$ is just a single leaf, setting $U^1 = U$ satisfies the desired properties.

Assume that for every proper subtree of $U$ the result holds. First, if $w(U) < \alpha$ then setting $U^1 = U$ again completes the proof. Thus, assume that $w(U) \geq \alpha$. Then we can traverse the tree $U$ starting at $r(U)$ and going to its leaves in the following way: if we are currently at node $u$, we go to the child of $u$ with greatest weight. We stop the traversal when all children of $u$ have weight less then $\alpha$. We argue that the node $u$ at the end of the traversal has one of the two properties:

(i) $u$ is a leaf of $U$. Because $u$ is the heaviest of its siblings (and the traversal has not ended at its father), it follows that $w(u) \geq \alpha$.

(ii) $w(u) \geq \alpha$ and all children of $u$ have weight less than $\alpha$.

If these are not the cases, then there is at least one child of $u$ with weight $\geq \alpha$, contradicting the stop of the traversal.

In any case we can use the inductive hypothesis on $U - U_u$ to find a partition $\{U^1, \ldots U^k\}$ for $U - U_u$. It can be readily verified that $\{U^1, \ldots, U^k, U_u\}$ is a partition of $U$ with the desired properties. $\square$

## B Proof of Theorem 1

By means of contradiction, suppose that Hypothesis 1 holds for a node $h$. We then find an assignment $A'$ better than the optimal assignment $A^*$, reaching the contradiction. We argue that without loss of generality we can assume that $w(T) \geq 1$. If $w(T)$ is less then one, then we multiply each of the weights by a constant and find a weight function $w'(T)$ such that $w(T)$. Notice that from the linearity of the objective function $A^*$ is an optimal assignment for $(T, w')$ and that Hypothesis 1 holds for it. Moreover, the linearity implies that the assignment $A'$ constructed during the proof is better than $A^*$ for both instances $(T, w')$ and $(T, w)$, which then proves the result for the original instance. Also without loss of generality we assume that $A^*$ is a non-crossing assignment and that it only contains proper hotlinks, that is hotlinks of the form $(u, v)$ where $v \neq u$ and $v$ is not a child of $u$.

During this section we use $\mathbf{T}_u$ as a shorthand for $\mathbf{T}_u(A^*)$, for any node $u$, and $\mathrm{OPT}(T')$ as the shorthand for $\mathrm{OPT}_1(T')$, for any tree $T'$.

Define $Q = (q_1 \to \ldots \to q_{|Q|})$ as the user path from $r$ to $h$ in $T + A^*$. Consider some node $q_i \neq h$ in the path $Q$. For each child $j$ of $q_i$ that does not belong to $Q$, we define the tree $T_i^j$ as $T_j - T_{q_{i+1}}$ (Figure 3). Let $D$ be the subset of nodes of the path $Q$ whose hotlinks in $A^*$ are not pointing to nodes in $Q$. The set $D$ also includes the nodes of $Q$ that do not have hotlinks in $A$. Let $S$ be the set of nodes in $T_h$ which are endpoints of hotlinks from nodes in $D$, namely $S = \{s \in T_h : (d, s) \in A \text{ for some } d \in D\}$. Notice that because $h \in Q$, this definition implies that $h \notin S$.

Let $\{H_1, \ldots, H_k\}$ be a partition of $T_h$ given by Lemma 3 with $U = T_h$ and $\alpha = (w(T) + 1)/|D|$. Notice that from the choice of $\alpha$, $k$ must be at most $|D|$. Henceforth, we assume that the trees $\{H_i\}$ are labeled such that for all $i < j$, $r(H_i)$ is not an ancestor of $r(H_j)$. Lastly, define $\sigma_j = S \cap H_i^j$ and $\overline{H}_i^j = H_i^j - (\bigcup_{s\in\sigma_j} \mathbf{T}_s)$.

We now are finally able to construct the assignment $A$. Let $A_q$ be the subset of hotlinks of $A^*$ with both endpoint in $Q$. Let $A_i^j$ be an optimal assignment for $T_i^j$, $\overline{A}_i^j$ and optimal assignment for $\overline{H}_i^j$ and $A_s$ an optimal assignment for $s \in S$ (we assume that each of these assignments are non-crossing). Let $d_i$ be the $i$th node of $D$ closest to $r$ (where 'closest' is taken with respect to $T$). The assignment $A$ is defined in three parts: the assignment for $T - (T_h - h)$ given by $A_{out} = A_q \cup (\bigcup_{i,j} A_i^j)$, the assignments for each of the trees $H_i^j$ given by $A_{H_i^j} = \overline{A}_i^j \cup (\bigcup_{s\in\sigma_j} (A_s \cup (j, s))$ and the 'coupling' assignment $A_d = \bigcup_i (d_i, r(H_i))$. Then $A$ is defined as $A = A_{out} \cup (\bigcup_{i,j} A_{H_i^j}) \cup A_d$. It follows from the ordering we imposed on the trees $\{H_i\}$ that $A$ is a non-crossing assignment.

The next lemmas present important properties of the structures defined above as well as bounds for the cost of the assignment $A$. In order to simplify the notation, we define $l_i = d(r, d_i, T + A^*)$.

**Lemma 7.** *If there is a hotlink $(v, u) \in A^*$ such that $v$ is a proper ancestor of $h$ and $u$ is a proper descendant of $h$, then $u \in S$.*

*Proof.* By means of contradiction, suppose that $v$ does not belong to the path $Q$. If that is the case, then there must be a hotlink $(h_1, h_2) \in A^*$ in $Q$ such that $h_1$ is a proper ancestor of $v$ and $h_2$ a proper descendant of $v$. Thus, there is a crossing between $(v, u)$ and $(h_1, h_2)$, which contradicts the assumption that $A^*$ is a non-crossing assignment. Consequently $v \in Q$ and by the definition of $S$ the node $u$ must belong to $S$. □

The next lemmas are consequence of Lemma 7 and can be easily verified:

**Lemma 8.** *For $s \in S$, all nodes of $T_s$ belong to $(\bigcup_{s'\in S} \mathbf{T}_s)$.*

**Lemma 9.** *The trees $\mathbf{T}_h$ and $\{\mathbf{T}_s\}_{s\in S}$ define a partition of nodes of $T_h$.*

**Lemma 10.** *Consider some $s \in S$. Then $\mathbf{T}_s$ is a subtree of some $H_i^j$.*

*Proof.* By means of contradiction suppose that $\mathbf{T}_s$ is not a subtree of some $H_i^j$. By definition $h \notin \mathbf{T}_s$. Suppose that $s \in H_i^j$; then because $\mathbf{T}_s$ is not a subtree of $H_i^j$, it must contain a node $u \notin H_i^j$. Because the trees $\{H_i\}$ contain all nodes of $T_h$, $u$ must belong to some tree $H_{i'}$. Furthermore, it is clear that $r(H_{i'})$ must be a proper descendant of $s$. Therefore, there is a path $(s \rightsquigarrow r(H_{i'}) \rightsquigarrow u)$ in $T$, and because $\mathbf{T}_s$ is a subtree of $T$ containing both $s$ and $u$, it must also contain $r(H_{i'})$.

In any of the cases where $s \in \{r(H_i)\}$ or $s \in \{H_i^j\}$, the tree $\mathbf{T}_s$ contains a node $r(H_i) \neq h$. Therefore $H_i \subseteq T_s$, so $w(H_i) \leq w(T_s)$. But from Lemma 8 $w(T_s) \leq \sum_{s'\in S} w(\mathbf{T}_{s'})$. Clearly the trees $\{\mathbf{T}_s\}$ are disjoint to $\mathbf{T}_h$, and therefore Hypothesis 1 implies that $\sum_{s'\in S} w(\mathbf{T}_{s'}) \leq w(T)/c \leq w(T)/|D|$. Consequently $w(H_i) \leq w(T)/|D|$, which contradicts its construction. □

The following lemma is a consequence of Lemmas 7 and 10:

**Lemma 11.** *$\overline{H}_i^j$ is a tree. Furthermore, either $\overline{H}_i^j$ does not contain any nodes or it is rooted at node $j$.*

**Lemma 12.** *$d(r, h, T + A) = d(r, h, T + A^*)$*

*Proof.* Recall that the path from $r$ to $h$ in $T + A^*$ is $Q$ and define $Q' = (q'_1 \to \ldots \to q'_{|Q'|})$ as the path from $r$ to $h$ in $T + A$. By induction assume that $q'_i = q_i$ (clearly this holds for the base case $r = q'_1 = q_1$). By the construction of $A$, $q_i$ has the same set of arcs and hotlinks in $T + A$ and in $T + A^*$. Consequently, a user in $q_i$ that is going to $h$ on the enhanced tree $T + A$ must chose $q_{i+1}$ as his next node, and thus $q'_{i+1} = q_{i+1}$. □

**Lemma 13.** *All nodes of $\mathbf{T}_h$ are contained in $(\bigcup_{i,j} \overline{H}_i^j) \cup (\bigcup_i r(H_i))$.*

The next lemmas come from the discussion presented in section 3:

**Lemma 14.** *$EP(T, A)_{T_i^j} \leq EP(T, A^*)_{T_i^j} + c \cdot w(T_i^j)$*

**Lemma 15.** *Consider $u \in H_i$. Then $d_i$ belongs to the user path from $r$ to $u$ in $T + A$.*

**Lemma 16.** *Consider any tree $H_i^j$. For any node $s \in \sigma_j$, $EP(T, A)_{\mathbf{T}_s} \leq EP(T, A^*)_{\mathbf{T}_s} + (l_i + 3)w(\mathbf{T}_s)$.*

**Lemma 17.** *$EP(T, A)_{\overline{H}_i^j} \leq EP(T, A^*)_{\overline{H}_i^j} + (l_i + 2 - c) \cdot w(\overline{H}_i^j)$*

**Lemma 18.** *$EP(T, A)_{r(H_i)} \leq EP(T, A^*)_{r(H_i)} + (l_i + 1 - c) \cdot w(r(H_i))$*

**Lemma 19.** *Consider a tree $H_i^j$. Then $\mathbf{T}_j(A) = H_i^j$*

*Proof.* Because $H_i^j$ is a tree, it must be the subtree of $T$ induced by its nodes. As $\mathbf{T}_j(A)$ is also the subtree of $T$ induced by its nodes, it suffices to show that both $\mathbf{T}_j(A)$ and $H_i^j$ contain the same nodes.

($\supseteq$) Using Lemma 15, it is easy to see that the path from $r$ to a node $u \in H_i$ is $(r \rightsquigarrow d_i \to r(H_i) \to j \rightsquigarrow u)$. Therefore all nodes of $H_i^j$ belong to $\mathbf{T}_j(A)$.

($\subseteq$) Clearly only nodes of $T_h$ can belong to $\mathbf{T}_j(A)$. So consider a node $u$ which belongs to both $T_h$ and $\mathbf{T}_j(A)$. By means of contradiction assume that $u \notin H_i^j$. Because $u$ needs to be a descendant of $j$ in order to be in $\mathbf{T}_j(A)$, it is easy to see that $u$ cannot belong to $H_i^{j'}$ for $j \neq j'$. As the trees $\{H_i\}$ define a partition of $T_h$, $u$ must be in some tree $H_{i'}$ different than $H_i$. Moreover, $r(H_{i'})$ must be a descendant of $j$. Again using Lemma 15, the path from $r$ to $u$ must be $(r \rightsquigarrow d_{i'} \to r(H_{i'}) \rightsquigarrow u)$. But if $r(H_{i'})$ is a descendant of $r(H_i)$, then $j$ cannot belong to this path from $r$ to $u$. This contradicts our choice of $u$ and completes the proof. □

**Lemma 20.** *There cannot be two consecutive hotlinks in $Q$*

*Proof.* By means of contradiction suppose there are two hotlinks $(u_1, u_2)$ and $(u_2, u_3)$ in $A^*$, such that $u_1, u_2$ and $u_3$ are consecutive nodes in $Q$ in this order.

Define $G = \mathbf{T}_{u_1}(A^*)$ and let $A_G$ be the hotlinks of $A^*$ with both endpoints in $G$. Notice that $G$ is rooted at $u_1$ and that (because $A^*$ was assumed to be non-crossing) $A_G$ contain the hotlinks $(u_1, u_2)$ and $(u_2, u_3)$.

Let $u_1'$ be the child of $u_1$ that belongs to the path from $u_1$ to $u_2$ in $T$ (because we have assumed that $A^*$ does not contain proper hotlinks $u_1'$ is an ancestor of $u_2$). Now define a new assignment $A_G'$ for $G$ in the following way: $A_G' = A_G - (u_1, u_2) \cup (u_1', u_2) \cup (u_1, u_3)$. Let us analyze the paths in $G + A_G$ and in $G + A_G'$.

Consider a node $u \in G_{u_3}$. It is easy to see that the path from $u_1$ to $u$ in $G + A_G$ is $(u_1 \to u_2 \to u_3 \rightsquigarrow u)$. On the other hand the path from $u_1$ to $u$ in $G+A_G'$ is $(u_1 \to u_3 \rightsquigarrow u)$. Because we have not added or removed hotlinks from nodes in $G_{u_3}$, the path $(u_3 \rightsquigarrow u)$ is the same in $G + A_G$ and $G + A_G'$. Therefore $d(u_1, u, G + A_G') = d(u_1, u, G + A_G) - 1$, and weighting over all $u \in G_{u_3}$ we have $\mathrm{EP}(G, A_G')_{G_{u_3}} = \mathrm{EP}(G, A_G)_{G_{u_3}} - w(G_{u_3})$.

Now consider a node $u \in G_{u_2} - G_{u_3}$. Similarly, the path from $u_1$ to $u$ in $G + A_G$ is $(u_1 \to u_2 \rightsquigarrow u)$ and in $G + A_G'$ is $(u_1 \to u_1' \to u_2 \rightsquigarrow u)$. Again, because we have not added or removed hotlinks to nodes in $G_{u_2}$ the path $(u_2 \rightsquigarrow u)$ is the same in $G + A_G$ and $G + A_G'$. Therefore weighting over all $u \in G_{u_2} - G_{u_3}$ we have $\mathrm{EP}(G, A_G')_{G_{u_2}-G_{u_3}} = \mathrm{EP}(G, A_G)_{G_{u_2}-G_{u_3}} + w(G_{u_2} - G_{u_3})$.

Now consider a node $u \in G - G_{u_2}$. Because we have not added or removed hotlinks with both endpoints in $G - G_{u_2}$, the path form $u_1$ to $u$ is the same in $G + A_G$ and $G + A_G'$. Consequently $\mathrm{EP}(G, A_G')_{G-G_{u_2}} = \mathrm{EP}(G, A_G)_{G-G_{u_2}}$. Because he above analysis contemplates all nodes of $G$, we have that $\mathrm{EP}(G, A_G') = \mathrm{EP}(G, A_G) - w(G_{u_3}) + w(G_{u_2} - G_{u_3})$.

By definition, in order for users to reach the nodes of $\mathbf{T}_h(A^*)$ they have to traverse $Q$, and consequently $u_3$. Therefore $\mathbf{T}_h(A^*) \subseteq G$. Furthermore this implies that $\mathbf{T}_h(A^*) \subseteq G_{u_3}$, and consequently $w(G_{u_3}) \geq \mathbf{T}_h(A^*)$. Using Hypothesis 1 it follows that $w(G_{u_3}) \geq w(T)(c-1)/c$ and $w(G_{u_2} - G_{u_3}) \leq w(T)/c$. Using the above relationship between $\mathrm{EP}(G, A_G')$ and $\mathrm{EP}(G, A_G)$ we have that $\mathrm{EP}(G, A_G') < \mathrm{EP}(G, A_G)$.

Now we can use Corollary 1 to replace the assignment $A_G$ by $A_G'$ in $A^*$. However, this leads to an improved assignment for $T$ which contradicts the optimality of $A^*$. □

**Lemma 21.** $\sum_{i=1}^{k} l_i w(H_i) \leq \frac{3.1c \cdot w(T_h) + 25w(T_h)}{4}$

*Proof.* The rough idea of the proof is the following. Superimpose the path $Q$ over the nonnegative part of the real line, where a node $i$ goes to the number $l_i$. Because there are no two consecutive hotlinks in $Q$ (Lemma 20), nodes of $D$ cannot be very concentrated close to $c$. Also, because almost all weights $w(H_i)$ need to be 'balanced', we have that $w(H_i) \approx w(T_h)/k$ for all $i$. The expression then reduces to $w(T_h) \sum_{i=1}^{k} l_i/k$. Viewing the expression $\sum_{i=1}^{k} l_i w(H_i)$ as the 'center of mass' of the nodes in $D$, and because they are not very concentrated close to $c$, it follows that $\sum_{i=1} k l_i w(H_i) \leq \alpha w(T_h)$, being $\alpha$ a constant less than one.

First notice that $l_1 < l_2 < \ldots < l_k$. We will try to maximize the function $f(\{w(H_i)\}, \{l_i\}) = \sum_{i=1}^{k} l_i w(H_i)$ by choosing the values of the $l_i$'s and $w(H_i)$'s, and then show that for even the best choice the desired upper bound holds. Let us find the values of the $w(H_i)$'s that maximize $f$.

As the $l_i$'s are monotonically increasing, it should be clear that the best strategy is to put $w(H_k)$ as large as possible. But there are two constraints limiting this value, namely the fact that the sum of the $w(H_i)$'s equals $w(T_h)$ and the fact that all but one of the $w(H_i)$'s are lower bounded by $(w(T) + 1)/|D|$. The first decision is then to chose which of the $H_i$'s will be the light one, that is the only one that $p(H_i)$ can be lower than $(w(T) + 1)/|D|$. Using the ordering on the $l_i$'s, it is easy to see that in order to maximize

$f$ we have to choose $H_1$ as the light subtree. For choosing the other $w(H_i)$'s we try to maximize the following (fractional) mathematical program:

$$\max\{f : w(H_i) \geq (w(T)+1)/|D|, \text{ for } i > 1; \sum_{i=1}^{k} w(H_i) = w(T_h)\}$$

or even better, the relaxed version

$$\max\{f : w(H_i) \geq w(T)/|D|, \text{ for } i > 1; \sum_{i=2}^{k} w(H_i) \leq w(T)\}$$

Again, it follows from the ordering on $l_i$ that the optimal solution is $w(H_i)^* = w(T)/|D|$, for $i = 2 \ldots k{-}1$ and $w(H_k)^* = (|D|{-}k{+}2)w(T)/|D|$. Thus, $f(\{w(H_i)^*\}, \{l_i\}) \geq f(\{w(H_i)\}, \{l_i\})$ for any other choice of the $w(H_i)$'s.

Now we need an upper bound on the $l_i$'s. Notice that because there cannot be two consecutive hotlinks in $Q$, it follows that $l_1 \leq 1$, $l_2 \leq 3$, …, $l_i \leq 2i-1 \leq 2i$. Furthermore, no $l_i$ can be greater than $c$, thus $l_i \leq \min\{2i, c\}$. Let $l_i^*$ be the values where $f$ attains its maximum value, which implies that $f(\{w(H_i)^*\}, \{l_i^*\})$ is an upper bound for $\sum_{i=1}^{k} l_i w(H_i)$. We can then proceed to the analysis, which is carried on two separate cases depending on a constant $0 \leq \beta < 1$ that is defined later.

**Case 1:** $l_k > \beta c$. Because $l_k \leq 2k$, the hypothesis implies that $k > \beta c/2$, and consequently $|D| > \beta c/2$. Upper bounding $f(\{w(H_i)^*\}, \{l_i^*\})$ we have:

$$f(\{w(H_i)^*\}, \{l_i^*\}) \quad \leq \quad \sum_{i=1}^{\beta c/2} 2i \cdot w(H_i)^* + \sum_{i=\beta c/2+1}^{k} c \cdot w(H_i)^*$$

Using the bound on $|D|$, the first term can be upper bounded by:

$$\sum_{i=1}^{\beta c/2} 2i \cdot w(H_i)^* = 2 \cdot \frac{1}{2} \cdot \frac{\beta c}{2}\left(\frac{\beta c}{2} + 1\right)\frac{w(T_h)}{|D|} = \frac{\beta^2 c^2 w(T_h)}{4|D|} + \frac{\beta c w(T_h)}{2|D|} \leq \frac{\beta^2 c^2 w(T_h)}{4|D|} + w(T_h)$$

The second term can be upper bounded by:

$$\begin{aligned}
\sum_{i=\frac{\beta c}{2}+1}^{k} c \cdot w(H_i)^* &\leq \sum_{i=\frac{\beta c}{2}+1}^{k}\left(\frac{c \cdot w(T_h)}{|D|}\right) + \frac{c(|D|-k+2)w(T_h)}{|D|} \\
&= \left(k - \frac{\beta c}{2}\right)\frac{c \cdot w(T_h)}{|D|} + \frac{(|D|-k+2)c \cdot w(T_h)}{|D|} = \left(|D| + 2 - \frac{\beta c}{2}\right)\frac{c \cdot w(T_h)}{|D|} \\
&= c \cdot w(T_h) + \frac{2c \cdot w(T_h)}{|D|} - \frac{\beta c^2 w(T_h)}{2|D|} \leq c \cdot w(T_h) + \frac{4w(T_h)}{\beta} - \frac{\beta c^2 w(T_h)}{2|D|}
\end{aligned}$$

Combining these bounds we have:

$$f(\{w(H_i)^*\}, \{l_i^*\}) \quad \leq \quad \frac{(\beta^2 - 2\beta)c^2 w(T_h)}{4|D|} + c \cdot w(T_h) + w(T_h) + \frac{4w(T_h)}{\beta}$$

Noticing that $(\beta^2 - 2\beta)$ is negative, we upper bound this term by taking the largest possible $|D|$, which is $c$:

$$\begin{aligned} f(\{w(H_i)^*\},\{l_i^*\}) &\leq \frac{(\beta^2-2\beta)c\cdot w(T_h)}{4} + c\cdot w(T_h) + w(T_h) + \frac{4w(T_h)}{\beta} \\ &= \frac{(\beta^2-2\beta+4)c\cdot w(T_h)}{4} + w(T_h) + \frac{4w(T_h)}{\beta} \end{aligned}$$

**Case 2:** $l_k \leq \beta c$. Upper bounding the $l_i$'s by $\beta c$, it follows that $f(\{w(H_i)^*\},\{l_i^*\}) \leq \beta c \cdot w(T_h)$.

In order to asymptotically (in terms of $c$) match the bounds on both cases, we can choose $\beta$ such that $\beta = \frac{\beta^2-2\beta+4}{4}$, which gives $\beta = 3-\sqrt{5}$. It is easy to verify that this guarantees the that $f(\{w(H_i)^*\},\{l_i^*\}) \leq \frac{3.1c\cdot w(T_h)+25w(T_h)}{4}$. □

**Conclusion of the proof of Theorem 1.** By definition, $\{\mathbf{T}_s\}_{s\in\sigma_j}$ and $\overline{H}_i^j$ form a partition of nodes of $H_i^j$. From the definition of $\{H_i\}$ it follows that $\{H_i^j\}$ and $\{r(H_i)\}$ form a partition of $T_h$. From Lemma 10 $S = \bigcup_j \sigma_j$. All these facts imply that $\{\mathbf{T}_s\}_{s\in S}$, $\{\overline{H}_i^j\}$ and $\{r(H_i)\}$ form a partition of $T_h$. Thus, employing the bounds from Lemmas 16-18 we have:

$$\begin{aligned} \mathrm{EP}(T,A)_{T_h} &= \sum_{s\in S}\mathrm{EP}(T,A)_{\mathbf{T}_s} + \sum_{i,j}\mathrm{EP}(T,A)_{\overline{H}_i^j} + \sum_i \mathrm{EP}(T,A)_{r(H_i)} \\ &\leq \mathrm{EP}(T,A^*)_{T_h} + \sum_i l_i w(H_i) + 3w(T) - c\left(\sum_{i,j} w(\overline{H}_i^j) + \sum_i w(r(H_i))\right) \end{aligned} \quad (11)$$

Lemma 13 implies that $\sum_{i,j} w(\overline{H}_i^j) + \sum_i w(r(H_i)) \geq w(\mathbf{T}_h)$, which from Hypothesis 1 is at least $\frac{(c-1)w(T)}{c}$. Using this fact and Lemma 21 to bound, respectively, the last and the second term of inequality (11) we have:

$$\mathrm{EP}(T,A)_{T_h} \leq \mathrm{EP}(T,A^*)_{T_h} + \frac{(41-0.9c)\cdot w(T)}{4} \quad (12)$$

Now we bound the cost of reaching the leaves of $\{T_i^j\}$. It is easy to see that $\{T_i^j\}$ form a partition of $T-(Q\cup T_h)$. Adding Lemma 14 for each $\{T_i^j\}$ leads to:

$$\mathrm{EP}(T,A)_{T-(Q\cup T_h)} = \sum_{i,j}\mathrm{EP}(T,A)_{T_i^j} \leq \mathrm{EP}(T,A^*)_{T-(Q\cup T_h)} + c\cdot w(T-(Q\cup T_h)) \quad (13)$$

Notice that $Q-h$ cannot contain any leaves, and therefore all of its nodes have zero weight. Thus $w(T-(Q\cup T_h))$ can be bounded by $w(T-T_h)$, which from Hypothesis 1 is at most $w(T)/c$. Substituting this bound on (13) we have:

$$\mathrm{EP}(T,A)_{T-(Q\cup T_h)} = \sum_{i,j}\mathrm{EP}(T,A)_{T_i^j} \leq \mathrm{EP}(T,A^*)_{T-(Q\cup T_h)} + w(T) \quad (14)$$

Again because $Q - h$ has only nodes with zero weight, the cost of reaching nodes of $T$ is the same as reaching nodes in $T - (Q \cup T_h)$ and in $T_h$, which from inequalities (12) and (13) can be bounded by:

$$\mathrm{EP}(T, A)_T = \mathrm{EP}(T, A)_{T-(Q\cup T_h)} + \mathrm{EP}(T, A)_{T_h} \le \mathrm{EP}(T, A^*) + \frac{(45 - 0.9c)w(T)}{4} \quad (15)$$

However, recall that $A$ has multiple hotlinks in roots of the trees $\{H_i^j\}$. Let $A_i^j$ be the hotlinks of $A$ with both endpoints in $H_i^j$. From the construction of $A$, it follows that $A_i^j$ has at most $|\sigma_j|$ hotlinks in $j$ and at most one hotlink in every other node. From Lemma 1 there is an assignment $A_i'^j$ for $H_i^j$ with at most one hotlink per node such that $\mathrm{EP}(H_i, A_i'^j) \le \mathrm{EP}(H_i, A_i^j) + |\sigma_j| w(H_i^j)$. In addition from Lemma 19 $H_i^j = \mathbf{T}_j(A)$, and therefore we can employ Corollary 1 to 'replace' the assignments $\{A_i^j\}$ by the assignments $\{A_i'^j\}$, building the final assignment $A'$. Using the relationship of the costs of the assignments $A_i^j$ and $A_i'^j$, Corollary 1 guarantees that:

$$\begin{aligned} \mathrm{EP}(T, A') &\le \mathrm{EP}(T, A) + \sum_{i,j} |\sigma_j| w(H_i^j) \le \mathrm{EP}(T, A) + \frac{w(T) + 1}{|D|} \cdot \sum_{i,j} |\sigma_j| \\ &= \mathrm{EP}(T, A) + \frac{(w(T) + 1)|S|}{|D|} \le \mathrm{EP}(T, A) + w(T) + 1 \end{aligned} \quad (16)$$

where the second inequality holds because by construction $w(H_i^j) \le (w(T) + 1)/|D|$, the next equality follows from Lemma 10 and the last inequality follows because the definition of $S$ implies that $|S| \le |D|$.

Therefore $A'$ is a valid assignment for $T$ and combining inequalities (15) and (16) we have:

$$\mathrm{EP}(T, A') \le \mathrm{EP}(T, A^*) + \frac{(49 - 0.9c)w(T)}{4} + 1$$

Recall that we have assumed $w(T) \ge 1$. For $c > 59$ we have $\mathrm{EP}(T, A') < \mathrm{EP}(T, A^*)$, which contradicts the optimality of $A^*$. Therefore, Hypothesis 1 does not hold and the theorem is true.

## C Proof of Theorem 2

In order to prove the theorem, it suffices to show that for every instance $(T, w)$ of the Hotlink Assignment Problem there is an optimal assignment $A^*$ such that the height of $T^{A^*}$ is bounded from above by $K(\log w(T) + \log n)$, for some constant $K$. As stated in section 3 we can then use the algorithm from [24] with $D = K(\log w(T) + \log n)$ to find an optimal assignment for $(T, w)$ in time $poly(n \cdot w(T))$.

First, we define $height(T + A)$ as the longest user path in $T + A$. Notice that this is the same as the height of the tree $T^A$.

**Lemma 22.** *Consider a tree $T$, an assignment $A$ and a node $u \in T$. Let $G = \mathbf{T}_u(A)$. Then for any $v \in G$, $\mathbf{G}_v(A) = \mathbf{T}_v(A)$.*

*Proof.* In order to prove the result it suffices to show that both $\mathbf{G}_v(A)$ and $\mathbf{T}_v(A)$ contain the same nodes, as both trees are defined as the subgraph of $T$ induces by their nodes.

The first observation is that Proposition 1 guarantees that the path from $u$ to a node $v \in G$ is the same in $G + A$ and $T + A$.

We start proving that $\mathbf{G}_v(A) \subseteq \mathbf{T}_v(A)$. Consider a node $y \in \mathbf{G}_v(A)$. By definition the path from $u$ to $y$ in $G + A$ (and consequently in $T + A$) contains $v$. But by definition of $G$ the path from $r$ to $y$ must contain $u$, and therefore the path form $r$ to $y$ in $T + A$ is $(r \rightsquigarrow u \rightsquigarrow v \rightsquigarrow y)$ and $v \in \mathbf{T}_v(A)$.

Now we prove that $\mathbf{G}_v(A) \supseteq \mathbf{T}_v(A)$. Consider a node $y \in \mathbf{T}_v(A)$. By definition, the path from $r$ to $y$ in $T + A$ contains $v$. But because $v \in \mathbf{T}_u(A)$, the path from $r$ to $v$ contains $u$ and therefore the path from $r$ to $y$ in $T + A$ is $(r \rightsquigarrow u \rightsquigarrow v \rightsquigarrow y)$. Clearly $v$ belongs to $G$ and because the path $(u \rightsquigarrow y)$ is the same in $T+A$ and $G+A$, $y$ also belongs $\mathbf{G}_v(A)$. $\square$

**Lemma 23.** *Consider a tree $T$ and an optimal non-crossing assignment $A^*$ for it. For every node $v \in T$ such that $d(r, v, T + A^*) \geq k \cdot c$, for an integer $k$, the following inequality holds: $w(\mathbf{T}_v(A^*)) \leq \left(\frac{c-1}{c}\right)^k w(T)$*

*Proof.* The proof goes by induction on $k$. Assume that for every $k' < k$ the following holds: for every node $v \in T$ such that $k' \cdot c \leq d(r, v, T + A^*) < (k' + 1)c$, we have $w(\mathbf{T}_v(A^*)) \leq \left(\frac{c-1}{c}\right)^{k'} w(T)$. Notice that result holds for the trivial base case $k' = 0$.

Now consider a node $v$ such that $k \cdot c \leq d(r, v, T + A^*) < (k + 1)c$. Let $u$ the node that belongs to the user path from $r$ to $v$ in $T + A^*$ such that $d(u, v, T + A^*) = c$ (such node exists because $k > k' \geq 0$). Clearly the distance from $r$ to $u$ in $T + A^*$ satisfies the inductive hypothesis and we have that $w(\mathbf{T}_u(A^*)) \leq \left(\frac{c-1}{c}\right)^{k-1} w(T)$

Let $G = \mathbf{T}_u(A^*)$ and define $A'$ as the hotlinks of $A^*$ with both endpoints in $G$. As $v$ belongs to $G$, Proposition 1 guarantees that $d(u, v, G + A') = d(u, v, T + A^*) = c$. In addition, because $A'$ is a subset of $A^*$, it is also clearly a non-crossing assignment. Hence Corollary 1 implies that $A'$ is an optimal assignment for $G$. Therefore, recalling that $G_v^{A'}$ has the same nodes as $\mathbf{G}_v(A')$ (and consequently the same weight) we can apply Theorem 1 to $G$ and have that:

$$w(\mathbf{G}_v(A')) \leq w(G)\left(\frac{c-1}{c}\right) \leq w(T)\left(\frac{c-1}{c}\right)^k \tag{17}$$

where the last inequality follows from the previous bound on $w(\mathbf{T}_u(A^*))$. By definition we have that $\mathbf{G}_v(A') = \mathbf{G}_v(A^*)$, which from Lemma 22 equals to $\mathbf{T}_v(A^*)$. Employing this last observation on inequality (17) we complete the proof. $\square$

Using the previous lemma we can complete the proof of Theorem 2. Define $U = \{u \in T : d(r, u, T + A^*) = K \log w(T)\}$. Notice that for a sufficiently large value of $K$, more specifically $K \geq 1/\log(c/(c-1))$, Lemma 23 guarantees that for every $u \in U$ we have $w(\mathbf{T}_u(A^*)) = 0$.

It was proved in [23] that for any tree $T'$ with $n'$ nodes, there is an assignment $A$ such that $\mathit{height}(T' + A)$ is upper bounded by $O(\log n')$. Then for each $u \in U$, we can find an assignment $A'_u$ such that $\mathit{height}(\mathbf{T}_u(A^*) + A'_u) \leq K \log n$. For each $u \in U$ let $A_u$ be the hotlinks of $A^*$ with both endpoints in $\mathbf{T}_u(A^*)$. Define $A' = A^* - (\bigcup_{u \in U} A_u) \cup (\bigcup_{u \in U} A'_u)$.

Because for every node $u \in U$ we have $w(\mathbf{T}_u(A^*)) = 0$, Corollary 1 implies that $A'$ is also an optimal assignment for $T$. Now we analyze the height of $T + A'$. Let $(r \rightsquigarrow v)$ be the longest path in $T + A'$. If $v \notin \bigcup_{u \in U} \mathbf{T}_u(A^*)$, then $d(r, v, T + A^*)$ must be less than $\log w(T)$. Because Lemma 4 guarantees that $d(r, v, T + A') = d(r, v, T + A^*)$, we have that $d(r, v, T + A')$ (and consequently the height of $T + A'$) is less than $\log w(T)$. Now suppose that $v \in \mathbf{T}_u(A^*)$, for some $u \in U$. By Lemma 4 $d(r, v, T + A') = d(r, u, T + A^*) + d(u, v, \mathbf{T}_u(A^*) + A'_u)$. Recall that $d(r, u, T + A^*) = K \log w(T)$ and the height of $\mathbf{T}_u(A^*) + A'_u$ implies that $d(u, v, \mathbf{T}_u(A^*) + A'_u) \leq K \log n$. Therefore, we have that $d(r, v, T + A')$ (and consequently the height of $T + A'$) is at most than $K(\log w(T) + \log n)$. In conclusion, we have found an optimal assignment $A'$ for $T$ such that $\mathit{height}(T + A') \leq K(\log w(T) + \log n)$, which implies that the dynamic programming algorithm solves the 1-HAP in $\mathit{poly}(n \cdot w(T))$ time.

To complete the proof of Theorem 2 that started in section 3, we need to argue that the solution for $(T, w')$ is an $(1 + \epsilon)$-approximation for $(T, w)$. Let $A^*$ be an optimal solution for $(T, w)$ and $A$ be the solution returned by the algorithm. Clearly for each node $u$ we have $K \cdot w'(u) \geq w(u)$. Therefore:

$$K \cdot \mathrm{EP}(T, A, w') = \sum_{u \in T} d(r, u, T + A) K \cdot w'(u) \geq \sum_{u \in T} d(r, u, T + A) w(u) = \mathrm{EP}(T, A, w) \quad (18)$$

Analogously, $K \cdot w'(u) \leq w(u) + K$ and hence $K \cdot \mathrm{EP}(T, A^*, w') \leq \mathrm{EP}(T, A^*, w) + \sum_u d(r, u, T + A^*) \cdot K$. Clearly the distance between any pairs of nodes in $T + A^*$ is at most $n$, thus $K \cdot \mathrm{EP}(T, A^*, w') \leq \mathrm{EP}(T, A^*, w) + n^2 \cdot K \leq \mathrm{EP}(T, A^*, w) + \epsilon \cdot W$. From the optimality of $A^*$ it follows that $K \cdot \mathrm{EP}(T, A, w') \leq \mathrm{EP}(T, A^*, w) + \epsilon \cdot W$. Because the optimal solution for $(T, w)$ is at least $W$ we have:

$$K \cdot \mathrm{EP}(T, A, w') \leq \mathrm{EP}(T, A^*, w) + \epsilon \cdot \mathrm{EP}(T, A^*, w) = (1 + \epsilon) \cdot \mathrm{OPT}(T, w) \quad (19)$$

By chaining inequalities (18) and (19) we complete the proof:

$$\mathrm{EP}(T, A, w) \leq K \cdot \mathrm{EP}(T, A, w') \leq (1 + \epsilon) \cdot \mathrm{OPT}(T, w) \quad (20)$$

## D Proofs for constant factor approximation

**Lemma 24.** *Consider a node $u$ in $\overline{T}_q$. Then, the user path from $r$ to $u$ in $T + A^*$ contains $c_q$.*

*Proof.* Consider a node $u \in \overline{T}_q$. By means of contradiction, assume that $u \notin \mathbf{T}_{c_q}(A^*)$. This implies there is a node $s \in T$ with the following properties: $s$ is an ancestor of $u$, $s$ is a proper descendant of $c_q$ and $(s', s) \in A^*$, where $s'$ is a proper ancestor of $c_q$. Let $(c_q, x)$ be the hotlink or arc from $c_q$ in $A^*$ such that $x \in \overline{T}_q$ (which must exist because $c_q$ captures $\overline{T}_q$). If $s$ is a descendant of $x$, then $s \in \overline{T}_q$, which contradicts the definition of $c_q$. On the other hand, if $s$ is an proper ancestor of $x$ we have a crossing in $A^*$ between $(s', s)$ and $(c_q, x)$, which contradicts the assumption that $A^*$ is a non-crossing assignment. □

*Proof of inequality* (4). The first observation is that a node $q \in Q$ can capture at most $k + 1$ different $\overline{T}_u$'s, because all of its arcs are arriving at the same $\overline{T}_u$. Therefore, a node $q$ can appear at most $k + 1$ times as the second parameter of the distance function on the summation on the left hand side. Let $w_q^i$ be the weight $w(\overline{T}_{q'})$ on the $i$th term that $q$ appears (we zero some of these weights accordingly if $q$ appears in less than $k+1$ of the terms). Then, we have that:

$$\sum_{q\in Q} d(r, c_q, T+A^*) w(\overline{T}_q) = \sum_{q\in Q} d(r,q,T+A^*) \cdot \sum_i w_q^i \tag{21}$$

Now we need to 'propagate' the weights $w_q^i$ to leaves. For each node $q \in Q$, add $k+1$ new leaves $q^i$; this new tree is denoted by $Q'$. Define the following weight function $w'$: for each new leaf $q^i$, let $w'(q^i) = w_q^i$ and let $w'(u) = 0$ for every other node of $Q'$ . Notice that $\{w'(u)\}_{u\in Q'} = \{w_q^i\}_{q\in Q,i} = \{w(\overline{T}_q)\}_{q\in Q}$. Also notice that the path from $r$ to $q^i$ in $Q' + A^*$ is exactly the path from $r$ to $q$ in $Q + A^*$ plus one hop from $q$ to $q^i$. Therefore:

$$\begin{aligned} \mathrm{EP}(Q', A^*, w') &= \sum_{q^i \in Q'} (d(r,q,T+A^*)+1) w'(q^i) = \sum_{q\in Q} (d(r,q,T+A^*)+1) \cdot \sum_i w'(q^i) \\ &= \sum_{q\in Q} d(r,q,T+A^*) \cdot \sum_i w_q^i + w(T) \end{aligned} \tag{22}$$

Define the normalized weight function $\widetilde{w}(q) = w'(q)/w'(Q')$ for each $q \in Q'$. Notice that $(Q', w')$ and $(Q', \widetilde{w})$ are instances of the $k$-Hotlink Assignment Problem, and by linearity of the objective function $\mathrm{EP}(Q', A^*, w') = \mathrm{EP}(Q', A^*, \widetilde{w}) \cdot w'(Q')$. Moreover, because $Q$ has degree at most $k$, even with the addition of the new leaves the tree $Q'$ has degree at most $2k+1$.

As discussed previously, the assignment $A^*$ defines a tree $Q^{A^*}$ and $\mathrm{EP}(Q', A^*, \widetilde{w})$ equals to the cost of reaching the leaves of $Q^{A^*}$ (when nodes are endowed with probabilities $\widetilde{w}$). Also, the degree of this tree is at most $2k+1+k = 3k+1$. Hence we can use Shannon's coding theorem to lower bound the cost of $\mathrm{EP}(Q', A^*, \widetilde{w})$:

$$\mathrm{EP}(Q', A^*, \widetilde{w}) \geq \mathrm{OPT}_k(Q', \widetilde{w}) \geq \frac{H(\{\widetilde{w}(q)\}_q)}{\log(3k+1)} = \frac{H(\{w'(q)\}_q)}{w'(Q')\log(3k+1)}$$

Because $k \geq 1$, it follows that $k^2 \geq k \Rightarrow k^2 + 2k + 1 \geq 3k+1 \Rightarrow (k+1)^2 \geq 3k+1$. Therefore, $\log(3k+1)) \leq 2\log(k+1)$. Using the relationship between $\mathrm{EP}(Q', A^*, w')$ and $\mathrm{EP}(Q', A^*, \widetilde{w})$, we have:

$$\mathrm{EP}(Q', A^*, w') \geq \frac{H(\{w'(q)\}_q)}{2\log(k+1)} = \frac{H(\{\overline{T}_q\})}{2\log(k+1)} \tag{23}$$

The result follows by chaining inequalities (21), (22) and (23) and applying the result on inequality (3). □

*Proof of inequality* (5). Let $L$ be the set of trees $T_q^j$ which are only single leaves and are adopted by $r$ in $A^*$, namely $L = \{T_q^j : T_q^j \text{ is a leaf and } (r,j) \in A^*\}$. Let $NL$ be the set of tree $T_q^j$ which are not single leaves and contain a node adopted by $r$, namely $NL = \{T_q^j : T_q^j \text{ is not a leaf and } (r,u) \in A^* \text{ for } u \in T_q^j\}$. Also, define $C$ as the set of trees $T_q^j$ that do not belong to $L$ or $NL$.

Consider a tree $T_r^j \in C$ (notice that $j$ is a child of $r$ in $T$). That user path in $T + A^*$ to reach a node $u \in T_r^j$ is $(r \to j \rightsquigarrow u)$ and thus $d(r,u,T+A^*) = 1 + d(j,u,T+A^*)$.

Adding this equality for all $u \in T_r^j$ and applying Lemma 2, we have that $\text{EP}(T, A^*)_{T_r^j} \geq w(T_r^j) + \text{OPT}_k(T_r^j)$.

Now for $q \neq r$, consider a tree $T_q^j \in C$. By assumption there cannot be an arc or hotlink in $T + A^*$ from $r$ to a node in $T_q^j$. Therefore, $c_q \neq r$ and from Lemma 24 the path from $r$ to a node $u \in T_q^j$ is $(r \rightsquigarrow c_q \rightsquigarrow u)$. These properties imply that $d(r, u, T + A^*) \geq 1 + d(c_q, u, T + A^*)$. Again adding this inequality for all $u \in T_q^j$ and then applying Lemma 2, we have that $\text{EP}(T, A^*)_{T_q^j} \geq w(T_q^j) + \text{OPT}_k(T_q^j)$.

Let $T_q^j$ be a tree in $L$. Because $T_q^j$ contains only one node, it is easy to see that $\text{EP}(T, A^*)_{T_q^j} = w(T_q^j) = w(T_q^j) + \text{OPT}_k(T_q^j)$. Finally, for a tree $T_q^j \in NL$ we can use Lemma 2 to derive the lower bound of $\text{EP}(T, A^*)_{T_q^j} \geq \text{OPT}_k(T_q^j)$.

Noticing that all leaves of $T$ are in some tree $T_q^j$, we can combine the previous lower bounds to attain the following bound for $\text{EP}(T, A^*)$:

$$\text{EP}(T, A^*) \geq \sum_{T_q^j \notin NL} w(T_q^j) + \sum_{T_q^j} \text{OPT}_k(T_q^j) \tag{24}$$

Consider a node $q$ such that some $T_q^j$ belongs to $NL$. Notice that if there is a tree $T_q^j \in NL$, them $q$ must have $k$ children in $Q$. Let $\{q_1, \ldots, q_k\}$ be the children of $q$ that also belong to $Q$. Also, let $k_q$ be the number of trees $\{T_q^j\}_j$ that belong to $NL$. Notice that $r$ has 'spent' at least $k_q$ hotlinks pointing to these trees $\{T_q^j\}_j$, and thus can use at most $k - k_q$ hotlinks to point to nodes in the tress $\{T_{q_i}\}_{i=1}^k$. As a consequence, at least $k_q$ of the trees $\{T_{q_i}\}_{i=1}^k$ do not have a node pointed by a hotlink from $r$. Let $NHL_q$ be the trees $\{T_{q_i}\}_{i=1}^k$ that do not have a node pointed by hotlinks from $r$ in the assignment $A^*$. Because each $q_i$ belongs to the heavy path $Q$, each tree in $NHL_q$ is at least as heavy as the trees $\{T_q^j\}$. As there are at least the same number of trees in $NHL_q$ than trees $\{T_q^j\}$ in $NL$:

$$\sum_{T' \in NHL_q} w(T') \geq \sum_{j : T_q^j \in NL} w(T_q^j) \Rightarrow \sum_{q : T_q^j \in NL} \sum_{T' \in HNL_q} w(T') \geq \sum_{T_q^j \in NL} w(T_q^j) \tag{25}$$

Now consider another node $q' \neq q$ such that there is a tree $T_{q'}^j$ in $NL$. Notice that a tree in $NHL_{q'}$ cannot be contained in a tree $T_{q_i} \in NHL_q$, otherwise $T_{q_i}$ would have to contain $T_{q'}$ and consequently a node of $T_{q'}^j$ pointed by a hotlink form $r$, which contradicts the definition of $NHL_q$. Because each tree in $\{NHL_q\}_q$ is maximal, for two trees in $\{NHL_q\}_q$ either one is contained in the other or they are disjoint. Combining these last two properties we have that the trees in $\{NHL_q\}_q$ must be pairwise disjoint. Moreover, this argument also implies that trees in $\{NHL_q\}_q \cup NL$ are pairwise disjoint.

As a consequence, the sum of the weights of the trees in $\{NHL_q\}_q \cup NL$ is at most $w(T)$, namely $\sum_{q : T_q^j \in NL} \sum_{T' \in NHL_q} w(T') + \sum_{T_q^j \in NL} w(T_q^j) \leq w(T)$. This fact combined with inequality (25) gives $\sum_{T_q^j \in NL} w(T_q^j) \leq w(T)/2$, or alternatively $\sum_{T_q^j \notin NL} w(T_q^j) \geq w(T)/2$ (recall that the trees $\{T_q^j\}$ contain all nodes of $T$ with non-zero weights). Applying this bound on inequality (24) completes the proof. $\square$

*Proof of Theorem 3.* Consider an instance $(T, w)$ of the $k$-Hotlink Assignment Problem. The proof goes by induction on the number of nodes of the input tree. Assume that for any tree $T'$ with fewer nodes than $T$ the algorithm outputs an $\alpha$-approximate hotlink assignment for $T'$, for some constant $\alpha$ that we make explicit later. For the base case of the induction, it should be clear that if $T'$ is just a leaf the hypothesis holds.

Now we argue that the result also holds for $T$. As each tree $T_i^j$ is properly contained in $T$, we can employ the inductive hypothesis in the upper bound given by (2):

$$\mathrm{EP}(T, A) \leq \frac{2H(\{w(\overline{T}_q)\})}{\log(k+1)} + \sum_{q \in Q} \sum_j \alpha \mathrm{OPT}_k(T_q^j) + w(T) \tag{26}$$

There are two cases that should be considered separately depending whether the value of $H(\{w(\overline{T}_i)\})/\log(k+1)$ dominates $w(T)$ or not. For simplicity of notation, we henceforth refer to $H(\{w(\overline{T}_i)\})$ as just $H$.

**High entropy case:** $H/\log(k+1) > 3w(T)$**.** From this hypothesis on the entropy, it follows that $-w(T) > -H/3\log(k+1)$. Substituting this inequality in the lower bound of equation (4), we have:

$$\mathrm{OPT}_k(T) > \frac{H}{6\log(k+1)} + \sum_{q \in Q} \sum_j \mathrm{OPT}_k(T_i^j) \tag{27}$$

Also from the entropy hypothesis $w(T) < H/3\log(k+1)$, and substituting in the upper bound (26) we have:

$$\mathrm{EP}(T, A) < \frac{7H}{3\log(k+1)} + \sum_{q \in Q} \sum_j \alpha \mathrm{OPT}_k(T_q^j) \tag{28}$$

By choosing $\alpha \geq 15$, inequalities (27) and (28) guarantee that $\mathrm{EP}(T, A) \leq 15 \cdot \mathrm{OPT}_k(T)$, thus concluding that inductive step for this case.

**Low entropy case:** $H/\log(k+1) \leq 3w(T)$**.** Combining this entropy hypothesis with the upper bound given by inequality (26), we have:

$$\mathrm{EP}(T, A) \leq 7w(T) + \sum_{q \in Q} \sum_j \alpha \mathrm{OPT}_k(T_q^j) \tag{29}$$

Again by choosing $\alpha \geq 15$, inequalities (5) and (29) guarantee that $\mathrm{EP}(T, A) \leq 15\mathrm{OPT}_k(T)$. This completes the proof of the theorem. $\square$

## E Proofs for binary search

Given any tree $T'$ and two nodes $u, v \in T'$, a lowest common ancestor (LCA) of $u$ and $v$ is a node $x$ such that: (i) $x$ is an ancestor of both $u$ and $v$ (ii) there is no proper descendant of $x$ which is an ancestor of both $u$ and $v$. The following proposition is easily verified:

**Proposition 2.** *Consider a tree $T'$ and two nodes of it $u$ and $v$ such that $u$ is neither an ancestor of $v$ nor a descendant of it. Then the lowest common ancestor $x$ of $u$ and $v$ is a proper ancestor of both of them. Furthermore, if $u'$ ($v'$) is the child of $x$ which is an ancestor of $u$ ($v$), then $u' \neq v'$.*

**Lemma 4.** *For every node or arc $v \in T'$, the node $u_v \in D^*$ is a descendant of $u(T')$.*

*Proof.* By means of contradiction suppose $u_v$ is not a descendant of $u(T')$. Let $v'$ be the arc or node of $T'$ corresponding to $u(T')$. The node $u_v$ cannot be a proper ancestor of $u(T')$, or that would contradict the choice of $u(T')$. Thus, from Proposition 2 $u(T')$ and $u_v$ are on different sides of their lowest common ancestor $u_{(i,j)}$. Without loss of generality assume that $u_v$ is a descendant of the right child of $u_{(i,j)}$ and that $u(T')$ is a descendant of the left child of $u_{(i,j)}$. By definition, $v$ belongs to $T_j$ and $v'$ belongs to in $T - T_j - (i,j)$. Thus, the root of $T'$ must be a proper ancestor of $j$ in order to include both $v$ and $v'$. But this implies that $(i,j) \in T'$. Because $u_{(i,j)}$ is a proper ancestor of $u(T')$, it is closer to the root of $D^*$ and contradicts the choice of $u(T')$. This completes the proof by contradiction. $\square$

**Lemma 25.** *Consider two different trees $T^j_{q_i}$ and $T^{j'}_{q_i}$. If $d(u(\overline{T}_{q_i}), u(T^j_{q_i}), D^*) = d(u(\overline{T}_{q_i}), u(T^{j'}_{q_i}), D^*)$, then $u(T^j_{q_i})$ and $u(T^{j'}_{q_i})$ are siblings.*

*Proof.* In order to simplify the notation, let $h_1 = u(T^j_{q_i})$ and $h_2 = u(T^{j'}_{q_i})$. Because the trees $T^j_{q_i}$ and $T^{j'}_{q_i}$ are disjoint, $h_1$ must be different than $h_2$. As both of them must be descendants of $u(\overline{T}_{q_i})$ (Lemma E), $h_1$ cannot be an ancestor or a descendant of $h_2$, otherwise their distances to $u(\overline{T}_{q_i})$ would be different. Therefore, Proposition 2 guarantees that $h_1$ and $h_2$ are on different sides of their lowest common ancestor $u_{(x,z)}$. Without loss of generality assume that $h_1$ is a descendant of the right child of $u_{(x,z)}$ and $h_2$ is a descendant of the left child of $u_{(x,z)}$.

By definition, the element corresponding to $h_1$ (which is an element in $T^j_{q_i}$) must be a descendant of $z$. However, $z$ cannot be an ancestor of $q_i$, otherwise both $h_1$ and $h_2$ would be at the right side of $u_{(x,z)}$. Therefore, $z$ belongs to $T^j_{q_i}$.

But the arc $(x,z)$ cannot belong to $T^j_{q_i}$, otherwise the fact that $u_{(x,z)}$ is a proper ancestor of $h_1$ would imply that $u_{(x,z)}$ is closer to $D^*$ and contradict the choice of $h_1$. As a consequence $z$ must be equal to the root of $T^j_{q_i}$, namely $q^j_i$. Then, the right child of $u_{(x,z)}$ must correspond to an arc of $T^{j'}_{q_i}$. This child must be $h_1$, otherwise this would again contradict the choice of $h_1$.

Because both $h_1$ and $h_2$ are descendants of $u(\overline{T}_{q_i})$, so is $u_{(x,z)}$. Then the paths from $u(\overline{T}_{q_i})$ to $h_1$ and $h_2$ are, respectively, $(u(\overline{T}_{q_i}) \rightsquigarrow u_{(x,z)} \rightarrow h_1)$ and $(u(\overline{T}_{q_i}) \rightsquigarrow u_{(x,z)} \rightsquigarrow h_2)$. In order to have $d(u(\overline{T}_{q_i}), h_1) = d(u(\overline{T}_{q_i}), h_2)$ the node $h_2$ must the the left child of $u_{(z,x)}$ and consequently $h_1$ and $h_2$ are siblings. $\square$

**Lemma 5.** *For every subtree $T'$ of $T$, $\sum_{v \in T'} d(u(T'), u_v, D^*) w(v) \geq OPT(T', w)$.*

*Proof.* The main idea is to construct a decision tree $D'$ for $T'$ by removing all nodes of $D^*$ which do not correspond to elements of $T'$. For that we define the recursive function $rec(u_v)$ which receives a node $u_v$ of $D^*$ and outputs a tree in the following way: if $v$ is a node or an arc of $T'$, compute $rec(r)$ and $rec(l)$ for the right and left child of $u_v$ (respectively $r$ and $l$) and return $u_v$ with $rec(r)$ appended as its right child and $rec(l)$ appended as its left child; if $v$ is not a node or an arc of $T'$ then return $rec(r)$ if $v$ is an ancestor of the root of $T'$ or return $rec(l)$ otherwise. The tree $D'$ is then defined as $D' = rec(u(T'))$ (Figure 12).

It follows from the construction that the tree $D'$ is binary. Also, we argue that $rec(u_{(i,j)})$ contains all nodes of $D^*_{u_{(i,j)}}$ which corresponds to elements of $T'$. If $(i,j) \in T'$

this follows directly by induction on the subtrees of $D^*_{u(i,j)}$; if $(i,j) \notin T'$ notice that only nodes in $D^*_r$ can correspond to elements of $T'$ (in case $j$ is an ancestor of $r(T')$) or only nodes in $D^*_l$ can correspond to elements of $T'$ (in case $j$ is not an ancestor of $r(T')$); the result then follows by induction. From Lemma E, $D^*_{u(T')}$ contains all nodes of $D^*$ corresponding to elements of $T'$, and therefore $D'$ also contains all nodes of $D^*$ corresponding to elements of $T'$.

Because $D'$ is basically a contraction of $D^*$, it is easy to see that for every node $u \in D^*_{u(T')}$ the path $(u(T') \rightsquigarrow u)$ in $D'$ is a subpath of the path $(u(T') \rightsquigarrow u)$ in $D^*$ (which is also easily proved by induction on the subtrees of $D^*$). Together with the previous discussion and the fact that $D^*$ is a valid decision tree, this implies that $D'$ is a valid decision tree for $T'$. This also implies that $d(u(T'), u, D') \leq d(u(T'), u, D^*)$ and consequently that $\text{cost}(D', w) \leq \sum_{v \in T'} d(u(T'), u_v, D^*)w(v)$. From the optimality we have that $\text{OPT}(T', w) \leq \text{cost}(D', w)$ and the result follows. $\square$

**Lemma 6.** *For some constant $c \geq 1$, $\sum_{q_i} d(r^*, u(\overline{T}_{q_i}), D^*)w(\overline{T}_{q_i}) \geq H(\{w(\overline{T}_{q_i})\})/c - w(T)$.*

*Proof.* Construct the tree $D'$ from $D^*$ as the following: for each node $q_i \in Q$, add a leaf $l_i$ to $u(\overline{T}_{q_i})$ (the relative position of siblings can be ignored in this analysis). Now define the probability distribution $w'$ for $D'$ such that $w'(l_i) = w(\overline{T}_{q_i})/w(T)$, and all other nodes have zero probability.

Clearly the distance from $r(D')$ to $l_i$ in $D'$ equals to $d(r, u(\overline{T}_{q_i}), D^*) + 1$. Then, the cost of $(D', w')$ is:

$$\text{cost}(D', w') = \sum_{l_i} d(r(D'), l_i, D')w'(l_i) = \sum_{q_i} \left(d(r^*, u(\overline{T}_{q_i}), D^*) + 1\right) \frac{w(\overline{T}_{q_i})}{w(T)}$$

Because the trees $\{\overline{T}_{q_i}\}$ are pairwise disjoint, for $q_i \neq q_j$ we have that $u(\overline{T}_{q_i}) \neq u(\overline{T}_{q_j})$. Therefore $D'$ is at most a ternary tree and Shannon coding theorem [9] implies that $\text{cost}(D', w') \geq H(\{w'(l_i)\})/c$ for some constant $c \geq 1$. Substituting this bound in the previous inequality and noticing that $H(\{w(\overline{T}_{q_i})\}) = H(\{w'(l_i)\}) \cdot w(T)$, we have:

$$\frac{H(\{w(\overline{T}_{q_i})\})}{c} \leq \sum_{q_i} \left(d(r^*, u(\overline{T}_{q_i}), D^*) + 1\right) w(\overline{T}_{q_i})$$

The result follows by reorganizing the previous inequality and noticing that $w(T) = \sum_{q_i} w(\overline{T}_{q_i})$ (because $\{\overline{T}_{q_i}\}$ define a partition of nodes of $T$). $\square$

**Lemma 26.** *$OPT(T, w) \geq \frac{w(T)}{2} + \sum_{q_i, j} OPT(T^j_{q_i}, w)$*

*Proof.* Consider a node $v \in T^j_{q_i}$. Applying Lemma E with $T' = T^j_{q_i}$ it follows that $d(r^*, u_v, D^*) = d(r^*, u(T^j_{q_i}), D^*) + d(u(T^j_{q_i}), u_v, D^*)$. Because $\{T^j_{q_i}\}$ and $\{q_i\}$ form a partition of nodes of $T$, the cost $\text{OPT}(T, w)$ can be written as:

$$\text{OPT}(T, w) = \sum_{q_i, j} d(r^*, u(T^j_{q_i}))w(T^j_{q_i}) + \sum_{q_i} d(r^*, u_{q_i})w(q_i) + \sum_{q_i, j} \sum_{v \in T^j_{q_i}} d(u(T^j_{q_i}), u_v)w(v) \quad (30)$$

First, notice that as $\{T^j_{q_{|Q|}}\}$ and $\{q_i\}$ do not contain any arcs, $\{u(T^j_{q_{|Q|}})\}$ and $\{u_{q_i}\}$ cannot be the root of $D^*$. Therefore, at most one distance $d(r^*, u(T^j_{q_i}))$ (for $q_i \neq q_{|Q|}$) of the first two summations of inequality (30) can be equal to zero, and all others must have value of at least one. But the construction of the heavy path guarantees that for $q_i \neq q_{|Q|}$ the weight $w(T^j_{q_i})$ is at most $w(T)/2$. As a consequence, the first two summations of inequality (30) can be lower bounded by $w(T)/2$. Combining this fact with a lower bound for the last summation provided by Lemma E (with $T' = T^j_{q_i}$), inequality (30) leads to the following bound:

$$\mathrm{OPT}(T, w) \geq \frac{w(T)}{2} + \sum_{q_i, j} \mathrm{OPT}(T^j_{q_i}, w)$$

□

The following theorem can be readily verified:

**Lemma 27.** *The computed tree $D$ is a valid decision tree for $T$.*

**Theorem 4.** *There is a linear time algorithm which provides a constant factor approximation for the problem of binary searching in trees.*

*Proof.* Consider a node $v \in T^j_{q_i}$. It is easy to see that the path from $r(D)$ to $u_v$ in $D$ is $(r(D) \rightsquigarrow u_{e^1_i} \rightsquigarrow r(D^j_i) \rightsquigarrow u_v)$. By construction, the path $(r(D) \rightsquigarrow u_{e^1_i})$ in $D$ is the same as the path $(r(D) \rightsquigarrow u_{q_i})$ in $D'$. Also, $d(u_{e^1_i}, r(D^j_i), D) = j$ and the path $(r(D^j_i) \rightsquigarrow u_v)$ is the same in $D$ and in $D^j_i$. Therefore, $d(r(D), u_v, D) = d(r(D), u_{q_i}, D') + j + d(r(D^j_i), u_v, D^j_i)$. In addition, the path from $r(D)$ to a node $q_i \in Q$ in $D$ can be written as $(r(D) \rightsquigarrow u_{e^1_i} \rightsquigarrow u_{q_i})$, which has length $d(r(D), u_{q_i}, D') + n_i$. Therefore, adding the (weighted) distance to reach nodes in $\{T^j_{q_i}\}$ and in $\{q_i\}$ we can find the cost of $D$:

$$\mathrm{cost}(D, w) = \mathrm{cost}(D', w') + \sum_j (j \cdot w(T^j_{q_i})) + \sum_{i,j} \mathrm{cost}(D^j_i, w) + \sum_{q_i} n_i w(q_i) \tag{31}$$

Now we upper bound $\mathrm{cost}(D', w')$. For each node $q_i \in Q$, define the weight function $\widetilde{w}(q_i) = w'(q_i)/w'(Q)$. In order to construct a decision tree for $(Q, w')$ in step (ii) of the algorithm, we actually construct a decision tree for the normalized instance $(Q, \widetilde{w})$. Let $D'$ be a decision tree for $(Q, \widetilde{w})$ given by the procedure of [6]. From their analysis, it follows that $\mathrm{cost}(D', \widetilde{w}) \leq H(\{\widetilde{w}(q_i)\}) + 2$. By linearity we have that $\mathrm{cost}(D', w') = \mathrm{cost}(D', \widetilde{w}) \cdot w'(Q) \leq (H(\{\widetilde{w}(q_i)\}) + 2) \cdot w'(Q) = H(\{w'(q_i)\}) + 2w'(Q)$. Noticing that $w'(Q) = w(T)$, we can use inequality (31) to provide the following upper bound on $\mathrm{cost}(D, w)$:

$$\mathrm{cost}(D, w) \leq H(\{w'(q_i)\}) + 2w(T) + \sum_{q_i, j} \left(j \cdot w(T^j_{q_i})\right) + \sum_{i,j} \mathrm{cost}(D^j_i, w) + \sum_{q_i} n_i w(q_i) \tag{32}$$

Now we start analyzing the lower bound for $\mathrm{OPT}(T, w)$. Notice that $\lfloor (j-1)/2 \rfloor \geq (j/2) - (3/2)$. Thus, recalling that $H(\{w'(q_i)\}) = H(\{w(\overline{T}_{q_i})\})$ and reorganizing inequality (9) we have:

$$\mathrm{OPT}(T, w) \geq \frac{H(\{w'(q_i)\})}{c} - \frac{5w(T)}{2} + \sum_{q_i, j} \frac{j \cdot w(T^j_{q_i})}{2} + \sum_{q_i, j} \mathrm{OPT}(T^j_{q_i}, w) + \sum_{q_i} n_i w(q_i) \tag{33}$$

We proceed by induction over the number of nodes of $T$, where the base case is the trivial one when $T$ has only one node. Notice that because each tree $T^j_{q_i}$ is properly contained in $T$, the inductive hypothesis implies that $D^j_i$ is an approximate decision tree for $T^j_{q_i}$, namely $\text{cost}(D^j_i, w) \leq \alpha\text{OPT}(T^j_{q_i}, w)$ for some constant $\alpha$. The analysis needs to be carried in two different cases depending on the value of the entropy.

**Case 1:** $H(\{w'(q_i)\})/c \geq 5w(T)$. From the entropy hypothesis it follows that $H(\{w'(q_i)\})/c - 5w(T)/2 \geq H(\{w'(q_i)\})/2c$. Applying this bound to inequality (33) we have:

$$\text{OPT}(T, w) \geq \frac{H(\{w'(q_i)\})}{2c} + \sum_{q_i,j} \frac{j \cdot w(T^j_{q_i})}{2} + \sum_{q_i,j} \text{OPT}(T^j_{q_i}, w) + \sum_{q_i} n_i w(q_i) \tag{34}$$

Also from the entropy hypothesis it is easy to see that $H(\{w'(q_i)\}) + 2w(T) \leq (5c + 2)H(\{w'(q_i)\})/5c$. Because $c \geq 1$, we have that $H(\{w'(q_i)\}) + 2w(T) \leq 2H(\{w'(q_i)\})$. Employing this bound on the first terms of inequality (32) and using the inductive hypothesis we have:

$$\text{cost}(D, w) \leq 2H(\{w'(q_i)\}) + \sum_{q_i,j} \left(j \cdot w(T^j_{q_i})\right) + \sum_{q_i,j} \alpha\text{OPT}(T^j_{q_i}, w) + \sum_{q_i} n_i w(q_i) \tag{35}$$

Setting $\alpha \geq 4c$ it follows from inequalities (34) and (35) that $\text{cost}(D, w) \leq \alpha\text{OPT}(T, w)$.

**Case 2:** $H(\{w'(q_i)\})/c \leq 5w(T)$. Applying the entropy hypothesis and the inductive hypothesis to inequality (32) we have:

$$\text{cost}(D, w) \leq (5c + 2)w(T) + \sum_{q_i,j} \left(j \cdot w(T^j_{q_i})\right) + \sum_{q_i,j} \alpha\text{OPT}(T^j_{q_i}, w) + \sum_{q_i} n_i w(q_i) \tag{36}$$

Now we need to find an appropriate lower bound for $\text{OPT}(T, w)$. For the first part, we use almost the same derivation that leads from (8) to (9); however, instead of using the entropy, the first summation of (8) is simply lower bounded by zero. This gives:

$$\begin{aligned} \text{OPT}(T, w) &\geq \sum_{q_i,j} \lfloor (j-1)/2 \rfloor w(T^j_{q_i}) + \sum_{q_i,j} \text{OPT}(T^j_{q_i}, w) + \sum_{q_i} n_i w(q_i) \\ &\geq -\frac{3w(T)}{2} + \sum_{q_i,j} \frac{j \cdot w(T^j_{q_i})}{2} + \sum_{q_i,j} \text{OPT}(T^j_{q_i}, w) + \sum_{q_i} n_i w(q_i) \end{aligned}$$

Adding $(\alpha - 1)$ times the inequality of Lemma 26 to the previous bound we have:

$$\alpha\text{OPT}(T, w) \geq \frac{(\alpha - 4)w(T)}{2} + \sum_{q_i,j} \frac{j \cdot w(T^j_{q_i})}{2} + \sum_{q_i,j} \alpha\text{OPT}(D^j_i, w) + \sum_{q_i} n_i w(q_i) \tag{37}$$

Setting $\alpha \geq 10c + 8$, inequalities (36) and (37) imply that $\text{cost}(T, w) \leq \alpha\text{OPT}(T, w)$. Therefore, the inductive step holds for both cases when $\alpha \geq 10c + 8$, which completes the proof of the theorem.

As mentioned in section 5, we can compute a heavy path decomposition in linear time and implement the algorithm in a straight forward way in $O(n \log n)$ time, where $n$ is the number of nodes of $T$. The bottleneck of the algorithm is sorting the trees $\{T^j_{q_i}\}$ at Step (iv). Now we present a linear time implementation using an 'approximate sorting'.

**Implementation in linear time.** Let $n$ be the number of nodes of $T$. Consider a node $q_i \in Q$ and let $W = \max_j\{w(T^j_{q_i})\}$. Define the sequence $WQ = (w_1, w_2, \dots, w_{n_i})$ as the weights $\{w(T^j_{q_i})\}$ in non-increasing order. For $0 \leq i < n$, let $w_{s_i}$ be the first element of $WQ$ which is not greater than $W/2^i$. For $0 \leq i \leq n$ we call the subsequence $(w_{s_i}, w_{s_i+1}, \dots, w_{s_{i+1}-1})$ as the $i$-block of $WQ$. The $n$-block is defined as $(w_{s_n}, \dots, w_{n_i})$. Notice that $WQ$ is the concatenation of these blocks.

Now we define a permutation $WQ' = (w_{\sigma(1)}, \dots, w_{\sigma(n_i)})$ where for $i < j$, is $w_{\sigma(i')} \in$ $i$-block and $w_{\sigma(j')} \in$ $j$-block, then $i' < j'$. That is $WQ' = (1\text{-block'}, 2\text{-block'}, \dots, n\text{-block'})$, where $i$-block' is a permutation of $i$-block. Notice that such permutation can be found in a single scan on all elements of $WQ$.

For every $0 \leq i < n$, it follows from the definition of $i$-block that:

$$\sum_{j=s_i}^{s_{i+1}-1} j \cdot w_j \geq \frac{W}{2^{i+1}} \sum_{j=s_i}^{s_{i+1}-1} j \tag{38}$$

It is easy to see that for every $s_i \leq j \leq s_{i+1} - 1$, for $0 \leq i \leq n$, $w_{\sigma(j)}$ is at most $W/2^i$. Therefore, for $0 \leq i < n$ we have:

$$\sum_{j=s_i}^{s_{i+1}-1} j \cdot w_{\sigma(j)} \leq \frac{W}{2^i} \sum_{j=s_i}^{s_{i+1}-1} j \tag{39}$$

In addition, because there are at most $n$ elements in $WQ'$, the sum $\sum_{j=s_n}^{n_i} j \cdot w_{\sigma(j)}$ can be upper bounded by $n^2(W/2^n)$. Therefore, for $n > 3$ this term can be upper bounded by $W$. Clearly $\sum_{j=1}^{n_i} j \cdot w_j \geq W$, thus:

$$\sum_{j=s_n}^{n_i} j \cdot w_{\sigma(j)} \leq \sum_{j=1}^{n_i} j \cdot w_j \tag{40}$$

Combining inequalities (38) to (40), we have:

$$\sum_{j=1}^{n_i} j \cdot w_{\sigma(j)} = \sum_{i=0}^{n-1} \sum_{j=s_i}^{s_{i+1}-1} j \cdot w_{\sigma(j)} + \sum_{j=s_n}^{n_i} j \cdot w_{\sigma(j)} \leq 2 \sum_{i=0}^{n-1} \sum_{j=s_i}^{s_{i+1}-1} j \cdot w_j + \sum_{j=1}^{n_i} j \cdot w_j \leq 3 \sum_{j=1}^{n_i} j \cdot w_j \tag{41}$$

Therefore, using the order given by $WQ'$ instead of the exact sort $WQ$ introduces a constant factor in second term of (36). It is straight forward to prove that this factor does not alter the guarantee of the algorithm. □