Bibliography

[AM01] M. Adler and B. Maggs. **Protocols for asymmetric communication channels**. *JCSS: Journal of Computer and System Sciences*, 63, 2001. 1

[PS93] R. de Prisco and A. de Santis. **On binary search trees**. *Inf. Process. Lett.*, 45(5):249–253, 1993. 6, 8, 8, 8.1, B.1

[BFN99b] Y. Ben-Asher, E. Farchi and I. Newman. **Optimal search in trees**. *SICOMP: SIAM Journal on Computing*, 28, 1999. 6, 6, 6

[BKL+03] P. Bose, D. Krizanc, S. Langerman and P. Morin. **Asymmetric communication protocols via hotlink assignments**. *Theory Comput. Syst.*, 36(6):655–661, 2003. 1

[PLS04b] 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. 1, 1, 4, A.3

[CPR+07] V. Chakaravarthy, V. Pandit, S. Roy, P. Awasthi and M. Mohania. **Decision trees for entity identification: approximation algorithms and hardness results**. In *PODS*, pages 53–62, 2007. 6

[Gallager68] R. Gallager. **Information theory and reliable communication**. John Wiley & Sons, Inc., New York, NY, USA, 1968. 5.2, B.2

[GKY02] M. Golin, C. Kenyon and N. Young. **Huffman coding with unequal letter costs**. In *STOC: ACM Symposium on Theory of Computing (STOC)*, 2002. 1

[DL06] 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. 1, 5, 5, 5.1, 8.5, A.4(a)

[Jacobs08] T. Jacobs. **An experimental study of recent hotlink assignment algorithms**. In *Proceedings of ALENEX*, 2008. 1

[Knight88] W. Knight. **Search in an ordered array having variable probe cost**. *SIAM J. Comput.*, 17(6):1203–1214, 1988. 6

[KPB99] R. Kosaraju, T. Przytycka and R. Borgstrom. **On an optimal split tree problem**. In *WADS: 6th Workshop on Algorithms and Data Structures*, 1999. 6

[LN04] E. Laber and L. Nogueira. **On the hardness of the minimum height decision tree problem**. *Discrete Applied Mathematics*, 144(1-2):209–212, 2004. 6

[MP04] 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. 1

[MOW08] 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. 6, 6, 6, 6

[NBB+00] G. Navarro, R. Baeza-Yates, E. Barbosa, N. Ziviani and W. Cunto. **Binary searching with nonuniform costs and its application to text retrieval**. *Algorithmica*, 27(2):145–169, 2000. 6

[PLS04] A. Pessoa, E. Laber and C. Souza. **Efficient implementation of hotlink assignment algorithms for web sites**. In *Proceedings of ALENEX*, 2004. 1, 1, 3

[BKK+00] 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. 1, 1, 5.2, 5.2

[GKM+03] 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. 1

[OP06] K. Onak and P. Parys. **Generalization of binary search: Searching in trees and forest-like partial orders**. In *FOCS*, pages 379–388, 2006. 6, 6, 6

[Jacobs07] 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. 1

[DL05] 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. 1, 1

[LA95] M. Lipman and J. Abrahams. **Minimum average cost testing for partially ordered components**. *IEEE Transactions on Information Theory*, 41(1):287–291, 1995. 6, 6

[SNB+03] J. Szwarcfiter, G. Navarro, R. Baeza-Yates, J. de S. Oliveira, W. Cunto and N. Ziviani. **Optimal binary search trees with costs depending on the access paths**. *Theor. Comput. Sci.*, 290(3):1799–1814, 2003. 6

[Karp61] R. Karp. **Minimum-redundancy coding for the discrete noiseless channel**. *IRE Trans. Inform. Theory*, 7:27–39, 1961. 1

[Knuth98] D. Knuth. **The art of computer programming, volume 3: sorting and searching**. Addison Wesley Longman Publishing Co., Inc., Redwood City, CA, USA, 1998. 6

[KKS01a] E. Kranakis, D. Krizanc and S. Shende. **Approximate hotlink assignment**. *Lecture Notes in Computer Science*, 2223:756–767, 2001. 1, A.4(a)

[PE97] M. Perkowitz and O. Etzioni. **Adaptive web sites: an AI challenge**. In *IJCAI (1)*, pages 16–23, 1997. 1

[LMP99] E. Laber, R. Milidiú and A. Pessoa. **Strategies for searching with different access costs**. In *ESA*, pages 236–247, 1999. 6

[LMP01] E. Laber, R. Milidiú and A. Pessoa. **On binary searching with non-uniform costs**. In *SODA*, pages 855–864, 2001. 6

[CDK+04] R. Carmo, J. Donadelli, Y. Kohayakawa and E. Laber. **Searching in random partially ordered sets**. *Theor. Comput. Sci.*, 321(1):41–57, 2004. 6, 6

[KGL+07] 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. 1

[CKK+03] J. Czyzowicz, E. Kranakis, D. Krizanc, A. Pelc and M. Martin. **Enhancing hyperlink structure for improving web performance**. *Journal of Web Engineering*, 1(2):93–127, March 2003. 1

A
Hotlink Assignment Problem

A.1 Preliminary lemmas

Proposition 4 *Consider a tree T rooted at node r and an assignment A for T. Also consider nodes $u, v \in T$ such that v is a proper descendant of u. Then the path P from r to v in $T + A$ has the form $(r \rightsquigarrow s \rightarrow s' \rightsquigarrow v)$, where s is an ancestor of u in T and s' is a descendant of u in T. Moreover, if $u \notin P$ then s and s' are respectively a proper ancestor and a proper descendant of u in T and $(s, s') \in A$.*

Lemma 15 (Local Change Lemma) *Consider some tree T rooted at r and a non-crossing assignment A for it. Let U be a subset 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 be another non-crossing 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 $\{\mathbf{T}_u(A)\}_{u \in U}$ or if $v \in U$, 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 $u \in U$. Also consider $x, y \in T$ such that x is a proper ancestor of u and y is a proper descendant of u. As $u \in \mathbf{T}_u(A)$, x and y cannot belong both to the same subtree of $T - \mathbf{T}_u(A)$. In particular, both x and y cannot belong to the same tree $\mathbf{T}_{u'}(A)$ with $u' \in U$, and consequently the hotlink (x, y) belongs to A if and only if it belongs to A'. The result then follows by the definition of the trees $\mathbf{T}_u(A)$ and $\mathbf{T}_u(A')$.

(ii) Notice that the path in $T + A$ from r to a node $v \notin \bigcup_{u \in U} \mathbf{T}_u(A)$ cannot contain a node $x \in \bigcup_{u \in U} \mathbf{T}_u(A)$, otherwise the path would be $(r \rightsquigarrow u \rightsquigarrow x \rightsquigarrow v)$ for some $u \in U$, implying $v \in \bigcup_{u \in U} \mathbf{T}_u(A)$. Then, all nodes in the path from r to v in $T + A$ have the same hotlinks in A' and in A. Then Proposition 2 implies that the path from r to v is the same in $T + A'$ and $T + A$. Notice that the path in $T + A$ from r to a node $u \in U$ cannot contain another

node $u' \in U$ different than u, otherwise this would imply that $u \in \mathbf{T}_{u'}(A)$ and contradict the fact that the trees $\{\mathbf{T}_u(A)\}_{u\in U}$ are pairwise disjoint. The result then holds for u by the same arguments as before.

(iii) Consider a node $u \in U$ and let v be a node in $\mathbf{T}_u(A)$. From (i) we have that $v \in \mathbf{T}_u(A')$ and consequently $d(r, v, T + A') = d(r, u, T + A') + d(u, v, T + A')$. From (ii) we have $d(r, u, T + A') = d(r, u, T + A)$. In addition, Proposition 1 asserts that $d(u, v, T + A') = d(u, v, \mathbf{T}_u(A) + A'_u)$ and the result holds. ■

Corollary 2 *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:*

$$EP(T, A') = EP(T, A) + \sum_{u\in U}(EP(\mathbf{T}_u(A), A') - EP(\mathbf{T}_u(A), A)) \qquad (1)$$

Proof: From Lemma 15, 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:

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

where the third equality follows from properties of Lemma 15.

By similar derivation, we have that:

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

Comparing the expressions for $\mathrm{EP}(T, A)$ and $\mathrm{EP}(T, A')$ leads to the result. ■

Lemma 16 *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 $\mathrm{EP}(\mathbf{T}_i(A), A'_i) \leq \mathrm{EP}(\mathbf{T}_i(A), A_i) + w(\mathbf{T}_i(A))$. 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:

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

■

Lemma 1 (Multiple Removal Lemma) *Consider a tree T rooted at node r and a weight function w. Let A be an assignment for T with at most g hotlinks leaving r and at most one hotlink leaving every other node. Then, there is an assignment A' with at most one hotlink per node such that $EP(T, A') \leq EP(T, A) + (g - 1)w(T)$.*

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 $\mathrm{EP}(T - T_v, A'_2) \leq \mathrm{EP}(T - T_v, A_2) + (g - 2) \cdot w(T)$. Now apply Lemma 16 to find an assignment A''_2 with no hotlink in r such that $\mathrm{EP}(T - T_v, A''_2) \leq \mathrm{EP}(T - T_v, A'_2) + w(T) \leq \mathrm{EP}(T - T_v, A_2) + (g - 1) \cdot w(T)$. 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 a tree T and a weight function w. Let T' be a subtree of T. If $v \in T$ is an ancestor of $r(T')$, then $\sum_{u \in T'} d(v, u, T + A)w(u) \geq OPT_g(T', w)$ 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 16 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. ■

Lemma 3 *Consider a tree U, a weight function w and a constant α. Then, there is a partition of U into subtrees such that each, except possibly the one containing $r(U)$, has weight with respect to w greater than α. In addition, for every tree U^i in the partition, each of the subtrees rooted at the children of $r(U^i)$ have weight not greater than α.*

Proof: The proof goes by induction on $w(U)$. If $w(U) \leq \alpha$ then setting $U^1 = U$ completes the proof. So assume that $w(U) > \alpha$. In addition, assume by induction that for every tree with weight less than $w(U)$ the result holds. Then we can traverse the tree U starting at $r(U)$ and going toward its leaves in the following way: if we are currently at node u, we go to the child v of u

with greatest $w(U_v)$. We stop the traversal when the subtrees of U rooted at the children of the current node have weight not greater than α.

We argue that, throughout the whole traversal, if u is the current node then $w(U_u) > \alpha$. Due to our assumption that $w(U) > \alpha$, this holds when $u = r(U)$. Now consider $u \neq r$; the claim must hold for u, otherwise because the trees rooted at the siblings of u have weight no more than $w(U_u)$, the traversal would have stopped in the previous step.

If u is the current node at the end of the traversal, than either u is a leaf of U or the subtrees of U rooted at the children of u have weight not greater than α. In any case, because $w(U_u) > \alpha \geq 0$ 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. ■

A.2 Proofs of lemmas used in Theorem 1

Proposition 3 *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.

Define $G = \mathbf{T}_{u_1}(A^*)$ and let A_G be the hotlinks of A^* with both endpoints in G. Clearly G is a tree rooted at node u_1. Moreover, because we have assumed that A^* is a non-crossing assignment and that no other hotlinks can point to nodes u_2 and u_3, it follows that there are no hotlinks in A^* departing from a proper ancestor of u_1 and pointing to proper descendants of u_1 which are ancestors of u_3. Consequently u_2 and u_3 belong to $\mathbf{T}_{u_1}(A^*) = G$ and A_G contains 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^* only contains proper hotlinks, u'_1 is a proper ancestor of u_2. Now we assume there is no hotlink in A^* departing from u'_1. If this is not the case, we can employ a combination of Lemma 16 and Corollary 2 to rebuild the assignment for $\mathbf{G}_{u'_1}(A^*)$ finding an assignment A'^* for T with no hotlink departing from u'_1. It is easy to see that this operations only introduces an additive factor of at most $w(G - G_{u_2})$ in the subsequent bounds and that this does not change the analysis.

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 user 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 $(u_1 \rightarrow u_2 \rightarrow u_3 \rightsquigarrow u)$. On the other hand the path from u_1 to u in $G + A'_G$ is $(u_1 \rightarrow u_3 \rightsquigarrow u)$. Because we have not added or removed hotlinks with

both endpoints in G_{u_3}, Proposition 1 implies that 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 $(u_1 \rightarrow u_2 \rightsquigarrow u)$ and in $G+A'_G$ is $(u_1 \rightarrow u'_1 \rightarrow u_2 \rightsquigarrow u)$. 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 the 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 nodes of $\mathbf{T}_h(A^*)$ in $T + A^*$ they have to traverse Q, and consequently u_1. Therefore $\mathbf{T}_h(A^*) \subseteq \mathbf{T}_{u_1}(A^*) = G$. Furthermore, such users also need to traverse node u_3, and hence the nodes of $\mathbf{T}_h(A^*)$ must be descendants of u_3. 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 $\mathrm{EP}(G, A_G)$ and the fact that $c > 2$, we have that $\mathrm{EP}(G, A'_G) < \mathrm{EP}(G, A_G)$.

Now we can use Corollary 2 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 5 *For any node $q \in Q$, the user path from r to q is the same in $T + A^1$ and in $T + A^*$.*

Proof: Consider a node $q \in Q$ and let u be an ancestor of h. Suppose that the hotlink (q, u) belongs to A^* (which implies that u is a descendant of q). A user going to node h in $T + A^*$ will necessarily traverse node q, so imagine this user at q. Because q do not have any other hotlink in A^*, it follows from the assumption that A^* only contains proper hotlinks that this user will take the hotlink (q, u) as his next step. Therefore, u must belong to Q and consequently the hotlink (q, u) also belongs to A^1. In addition, notice that q do not have any other hotlink in A^1. Then using Proposition 2, it follows that the path from r to h is the same in $T + A^*$ and $T + A^1$. Let this path be denoted by P. Noticing that the path from r to any node $q \in Q$ in both $T + A^*$ and $T + A^1$ is a subpath of P, the result follows. ■

Lemma 17 *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 Proposition 4 implies that 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. Notice that as $h_2 \in Q$, it must be a proper ancestor of u. Thus, there is a crossing between (v, u) and (h_1, h_2), which contradicts the assumption that A^* is a non-crossing assignment. Therefore $v \in Q$ and, because $u \notin Q$, it follows that $v \in D$ and consequently $u \in S$. ■

Lemma 18 *For $s \in S$, all nodes of T_s belong to $(\bigcup_{s' \in S} \mathbf{T}_{s'})$.*

Proof: Consider a node $s \in S$. The proof goes by induction on the distance from r to s in T. The property trivially holds when s is a leaf of T. Now assume that the property holds for all nodes in S which are proper descendants of s. Consider a node x in T_s but not in $\mathbf{T}_s$. As A^* is a non-crossing assignment, Definition 1-(ii) implies that x must belong to some tree T_u where $(v, u) \in A^*$, v is a proper ancestor of s and u is a proper descendant of s. As $s \in S$, there must be a hotlink (h_1, s) in A^* where h_1 is an ancestor of h. In order for A^* to be non-crossing, v must then be an ancestor of h_1 and consequently a proper ancestor of h. From Lemma 17 $u \in S$ and by the inductive hypothesis all nodes of T_u belong to $(\bigcup_{s' \in S} \mathbf{T}_{s'})$, hence so does x. Then each node of T_s either belongs to $\mathbf{T}_s$ or to $(\bigcup_{s' \in S} \mathbf{T}_{s'})$ and the inductive step is complete. ■

Lemma 19 *The trees $\mathbf{T}_h$ and $\{\mathbf{T}_s\}_{s \in S}$ define a partition of nodes of T_h.*

Proof: Combining Definition 1-(ii) and Lemma 17 it follows that $\mathbf{T}_h = T_h - (\bigcup_{s \in S} T_s)$. Therefore the nodes of T_h are either in $\mathbf{T}_h$ or $(\bigcup_{s \in S} T_s)$. From Lemma 18 it follows that all nodes of $(\bigcup_{s \in S} T_s)$ belong to $(\bigcup_{s \in S} \mathbf{T}_s)$, and therefore the nodes of T_h are either in $\mathbf{T}_h$ or in $(\bigcup_{s \in S} \mathbf{T}_s)$. Because $\mathbf{T}_h$ and $\{\mathbf{T}_s\}_{s \in S}$ are subtrees of T_h, all of their nodes belong to T_h and it suffices to show that these subtrees are pairwise disjoint in order to conclude the proof.

Again as $\mathbf{T}_h = T_h - (\bigcup_{s \in S} T_s)$, we have that $\mathbf{T}_h$ is disjoint to any tree T_s with $s \in S$. As $\mathbf{T}_s \subseteq T_s$, it follows that $\mathbf{T}_h$ is disjoint to any tree $\mathbf{T}_s$ with $s \in S$. Now consider two nodes $s \neq s' \in S$. If s is neither an ancestor nor a descendant of s' then clearly $\mathbf{T}_s$ is disjoint to $\mathbf{T}_{s'}$. So without loss of generality assume that s is a proper ancestor of s'. Due to the definition of S, there must be a hotlink (u, s') in A^* with u being an ancestor of h, and consequently a proper ancestor of s. It then follows by Definition 1-(ii) that $T_{s'}$ and $\mathbf{T}_s$ are disjoint and so are $\mathbf{T}_{s'}$ and $\mathbf{T}_s$. Therefore all trees $\mathbf{T}_h$ and $\{\mathbf{T}_s\}_{s \in S}$ are pairwise disjoint and the lemma follows. ■

Lemma 6 *Consider a node $s \in S$. Then $\mathbf{T}_s$ is a subtree of a tree H_i^j.*

Proof: By means of contradiction suppose that $\mathbf{T}_s$ is not a subtree of some H_i^j. By definition of S, $h \notin \mathbf{T}_s$. 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 $u \notin H_i^j$. Notice that u cannot belong to a tree $H_i^{j'}$ with $j' \neq j$, otherwise it would not belong to $\mathbf{T}_s$. Because the trees $\{H_i\}$ contain all nodes of T_h, then u must belong to some tree $H_{i'}$ with $i' \neq i$. Furthermore, as $s \notin H_{i'}$ and $u \in T_s$ it follows 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 $r(H_{i'})$.

As the subtrees $\{H_i^j\}$ and $\{r(H_i)\}$ form a partition of T_h, s must belong to either one of these subtrees. 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 18 $w(T_s) \leq \sum_{s' \in S} w(\mathbf{T}_{s'})$. As the trees $\{\mathbf{T}_{s'}\}_{s' \in S}$ and $\mathbf{T}_h$ are pairwise disjoint, Hypothesis 1 implies that $\sum_{s' \in S} w(\mathbf{T}_{s'}) \leq w(T)/c \leq w(T)/|D|$. As $w(T) > 0$, chaining previous observations we have that $w(H_i) \leq w(T)/|D| < 4w(T)/|D|$, which contradicts the construction of the tree H_i. ■

Lemma 20 *$\overline{H}_i^j$ is a tree. Furthermore, either $\overline{H}_i^j$ does not contain any nodes or it is rooted at node j.*

Proof: If $\overline{H}_i^j$ contains at most one node, then the result trivially holds. So consider two nodes $u, v \in \overline{H}_i^j$ such that u is a proper ancestor of v. By means of contradiction, assume there is no path in $\overline{H}_i^j$ between u and v. Notice there is a path from u to v in H_i^j, as the latter is a tree. Then there must be a node $s \in S$ which is a proper ancestor of v and a proper descendant of u. As a consequence $v \in T_s$, and from Lemma 18 $v \in \mathbf{T}_{s'}$ for some $s' \in S$. Notice however that s' must belong to the tree H_i^j. By construction $\overline{H}_i^j \cap \mathbf{T}_{s'} = \emptyset$ and hence v cannot belong to $\overline{H}_i^j$, which contradicts our choice of v.

For the second part of the lemma, suppose that j is not the root of $\overline{H}_i^j$. It follows by the definition of $\overline{H}_i^j$ that j must belong to S. Lemma 6 states that each of the trees $\{\mathbf{T}_s\}_{s \in S}$ is fully contained in one H_i^j. Because the trees $\{H_i^j\}$ are pairwise disjoint, this implies that for any node $s \in S - H_i^j$ the tree $\mathbf{T}_s$ is disjoint to H_i^j, and therefore we can write $\overline{H}_i^j$ as $\overline{H}_i^j = H_i^j - (\bigcup_{s \in S} \mathbf{T}_s)$. However, as $j \in S$ Lemma 18 implies that all nodes of T_j, and consequently of H_i^j, belong to $\bigcup_{s \in S} \mathbf{T}_s$. Therefore $\overline{H}_i^j$ cannot contain any nodes and the result follows. ■

Lemma 8 *Consider a tree H_i and a node $u \in H_i$. Then the user path in $T+A^2$ from r to u contains the node d_i.*

Proof: By means of contradiction, consider a node $u \in H_i$ for which the above property does not hold. Because A^2 is a non-crossing assignment, Definition 1-(i) implies that $u \notin \mathbf{T}_{d_i}(A^2)$ and consequently Definition 1-(ii) implies there is a hotlink (x, y) in A^2 such that: x and y are respectively a proper ancestor and a proper descendant of d_i and y is an ancestor of u. From the construction of the assignment A^2 we have that, for some $j < i$, x must be the node $d_j \in D$ and y the node $r(H_j)$. Because H_i and H_j are disjoint trees, $r(H_j)$ being an ancestor of u implies that $r(H_j)$ is an ancestor of $r(H_i)$. However, this contradicts the ordering on the trees $\{H_{i'}\}$ assumed at their definition, thus completing the proof. ■

Lemma 21 *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 8, it is easy to see that the path from r to a node $u \in H_i^j$ in $T + A^2$ is $(r \rightsquigarrow d_i \rightarrow r(H_i) \rightarrow j \rightsquigarrow u)$. But due to the discussion presented during the construction of the assignment A', the path from r to u is the same in $T + A^2$ and $T + A'$. Therefore all nodes of H_i^j belong to $\mathbf{T}_j(A')$.

($\subseteq$) Consider a node $u \in \mathbf{T}_j(A')$. Again, as the path from r to u is the same in $T + A'$ and $T + A^2$, it follows that $u \in \mathbf{T}_j(A^2)$. Clearly u also belongs to T_h. 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 $\mathbf{T}_j(A^2)$, 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, because $j \notin H_{i'}$, $u \in H_{i'}$ and $u \in T_h$, we have that $r(H_{i'})$ must be a proper descendant of j. Again using Lemma 8, the path in $T + A^2$ from r to u is $(r \rightsquigarrow d_{i'} \rightarrow r(H_{i'}) \rightsquigarrow u)$. But as $d_{i'}$ is a proper ancestor and $r(H_{i'})$ is a proper descendant of j, then j cannot belong to this path from r to u. This contradicts our choice of u and completes the proof. ■

A.3 Proof of Lemma 4

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

Proof: In order to prove the result it suffices to show that both $\mathbf{U}_v(A)$ 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 $v \in U$ is the same in $U + A$ and $T + A$.

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

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

Lemma 23 *Consider an instance (T, w) of the 1-HAP with T rooted at r and w an integer valued weight function. Let A^* be a non-crossing optimal assignment for this instance. In addition, consider the constant c given by Theorem 2. Then 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: First we remark that the previous discussions guarantee that such an optimal assignment that is non-crossing always exits.

The proof goes by induction on k. Assume that for every $0 \leq k' < k$ 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 this clearly 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 be 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 $k > 0$). 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 $U = \mathbf{T}_u(A^*)$ and define A' as the hotlinks of A^* with both endpoints in U. As v belongs to $\mathbf{T}_u(A^*) = U$, Proposition 1 guarantees that $d(u, v, U + A') = d(u, v, T + A^*) = c$. In addition, because A' is a subset of A^*, it is also clearly a non-crossing assignment. Hence, the optimality of A^* and Corollary 2 imply that A' is an optimal assignment for U. In addition, recall that $U_v^{A'}$ has the same nodes as $\mathbf{U}_v(A')$ and consequently the same weight. Then we can apply Theorem 2 to U and have that:

$$w(\mathbf{U}_v(A')) \leq w(U) \left(\frac{c-1}{c}\right) \leq w(T) \left(\frac{c-1}{c}\right)^k \tag{2}$$

where the last inequality follows from the previous bound on $w(\mathbf{T}_u(A^*)) = w(U)$.

Because A' is the subset of A^* with both endpoints in U we have that $\mathbf{U}_v(A') = \mathbf{U}_v(A^*)$, which from Lemma 22 equals to $\mathbf{T}_v(A^*)$. Employing this last observation on inequality (2) completes the proof. ■

Using the previous lemma we can conclude the proof of Lemma 4. Let A^* be a non-crossing optimal assignment for T. For some constant G we define $U = \{u \in T : d(r, u, T + A^*) = G \cdot \log w(T)\}$. Because w is integer valued, for a sufficiently large value of G (more specifically $G \geq 1/\log(c/(c-1))$) Lemma 23 guarantees that for every $u \in U$ we have $w(\mathbf{T}_u(A^*)) = 0$.

It was proved in [PLS04b] that for any tree T' with n' nodes, there is an assignment A such that $height(T' + A)$ is upper bounded by $O(\log n')$. Then for each $u \in U$, we can find an assignment A'_u such that $height(\mathbf{T}_u(A^*)+A'_u) \leq O(\log n)$. For each $u \in U$ let A_u be the hotlinks of A^* with both endpoints in $\mathbf{T}_u(A^*)$. We then replace the assignments A_u for the now assignments A'_u, that is, we 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 2 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 $T + A'$ (starting at node r). If $v \notin \bigcup_{u\in U} \mathbf{T}_u(A^*)$, then the path in $T + A^*$ from r to v does not contain any node in U and consequently $d(r, v, T + A^*)$ must be less than $G \log w(T)$. Because Lemma 15 guarantees that $d(r, v, T+A') = d(r, v, T+A^*)$, we have that $d(r, v, T + A')$ (and consequently $height(T + A')$) is less than $G \cdot \log w(T)$. Now suppose that $v \in \mathbf{T}_u(A^*)$, for some $u \in U$. By Lemma 15, $d(r, v, T + A') = d(r, u, T + A^*) + d(u, v, \mathbf{T}_u(A^*) + A'_u)$. Recall that $d(r, u, T + A^*) = G \log w(T)$ and that the construction of A'_u implies that $d(u, v, \mathbf{T}_u(A^*) + A'_u) \leq K \log n$ for some constant K. Therefore, we have that $d(r, v, T+A')$ (and consequently $height(T+A')$) is at most $G \log w(T)+K \log n$.

In any case, we have proved that there is an optimal assignment A' for (T, w) such that $height(T^{A'}) = height(T + A')$ is $O(\log w(T) + \log n)$, thus proving the result.

A.4 Constant factor approximation for the k-Hotlink Assignment Problem

(a) Solving k-HAP with internal weights

Approximate no-leaf assignment

In this section we present an algorithm for the k-HAP that given an instance (T, w) finds an assignment with cost at most $\frac{2H(w)}{\log(k+1)} + 2w(T)$ such

that no hotlink points to leaves of T (we call such assignment as a *no-leaf assignment*). This is later employed to devise an approximation for k-HAP with positive weights on internal nodes of T. Although the presentation uses the notation introduced in this work, the algorithm itself is a straightforward modification of the algorithm presented in [DL06]: when the original algorithm choses to add a hotlink pointing to a leaf u of T, the modified version adds a hotlink pointing to the parent of u. Hence, an adaptation of the argument employed in [DL06] can be used to prove the guarantee of our algorithm. Nonetheless, for sake of completeness we present a self-contained proof in the sequel.

Consider an instance (T, w) of the k-HAP, where T is a tree rooted at node r. The algorithm works recursively. It first finds a balanced partition $\{U^1, \ldots, U^p\}$ of T given by Lemma 3 and then adds hotlinks from r to the roots the trees $\{U^i\}$. Then, it recursively computes an assignment for each of the trees in the partition of T. The algorithm avoids adding hotlinks which point to leaves of T by computing the balanced partition with respect to an auxiliary weight function w', which essentially guarantees that the tree $\{U^i\}$ are not leaves of T. The algorithm is as follows:

(1) If $w(T) = 0$ or if T is a single leaf, then return the empty assignment.

(2) Define w' such that: if u is a leaf of T then $w'(u) = 0$; if u is an internal node of T then $w'(u)$ equals $w(u)$ plus the weights of the children of u which are leaves of T.

(3) Let $\{U^1, \ldots, U^p\}$ be a partition of T with respect to the weights w' given by Lemma 3 with $\alpha = w(T)/(k+1)$ (w.l.o.g we assume that $r(U^p) = r$).

(4) Recursively find an assignment A_i for each of the instances (U^i, w) with $1 \leq i < p$.

(5) If C is the set of children of r in U^p, find recursively an assignment A^p_i for each of instances (U^p_i, w) with $i \in C$.

(6) Return $A = \bigcup_{i=1}^{p-1} ((r, r(U^i)) \cup A_i) \cup \bigcup_{i \in C} A^p_i$.

In order to analyze the algorithm, we assume that $w(T) > 0$ and that T is not a single leaf, as otherwise it has a trivial behavior. It is easy to see that $w(T) = w'(T)$. By construction $w'(U^i) > w(T)/(k+1)$ for $i \leq p-1$, and hence $w(T) = w'(T) > (p-1)w(T)/(k+1)$. It then follows that $p-1$ cannot be greater than k. As a consequence, the assignment returned by the algorithm has at most k hotlinks departing from r. Due to the recursive nature of the algorithm, it follows by induction on the subtrees of T that the assignment returned by the algorithm contains at most k hotlinks departing from each node of T. In

addition, because for each leaf u of T we have $w'(u) = 0 \leq w(T)/(k+1)$, it follows that no tree U^i with $i \neq p$ can be a single leaf of T. This implies that the assignment returned by the algorithm does not contain any hotlinks from r which points to leaves of T. Again by induction, it can be easily seen that the assignment returned by the algorithm does not contain any hotlink pointing to leaves of T. Therefore, it returns a valid no-leaf k-assignment for T.

For the analysis of the approximation guarantee of the algorithm we need to introduce some additional notation. If during the execution of the algorithm over the instance (T, w) there is a recursive call to a subtree $T' \subseteq T$ rooted at node i, then we denote the tree T' by $\mathbf{T}_i(A)$, or $\mathbf{T}_i$ for short. It is easy to see that for each node u of T there is exactly one recursive to a tree rooted at u. (In addition, it can be proved that this notation is also consistent with our previous definition of $\mathbf{T}_i(A)$.)

For any subtree $T' \subseteq T$, define $\text{ALGO}(T')$ as the expected path of $T' + A'$ under w, where A' is the assignment returned by the execution of the algorithm over T'. In addition we define $ch(T)$ as the children of $r(T)$ in $T + A$, namely the union of the children of $r(T)$ in U^p and the endpoints of hotlinks from $r(T)$ in A, which are the nodes $\{r(U^i)\}_{i=1}^{p-1}$. In general, consider a tree $\mathbf{T}_i$ with its partition $\{U'^1, \ldots, U'^{p'}\}$ found in Step (3) of the execution of the algorithm over $\mathbf{T}_i$. Then the set $ch(\mathbf{T}_i)$ is defined as the children of $r(U'^p)$ in U'^p plus the nodes $\{r(U'^j)\}_{j=1}^{p-1}$.

Due to the recursive nature of the algorithm, it is easy to see that:

$$\text{ALGO}(T) = w(T) + \sum_{i \in C} \text{ALGO}(\mathbf{T}_i) + \sum_{i=1}^{p-1} \text{ALGO}(U^i) = w(T) + \sum_{i \in ch(T)} \text{ALGO}(\mathbf{T}_i)$$

For any leaf i of T we have $\text{ALGO}(\mathbf{T}_i) = 0$. So if L is the set leaves of T, the previous inequality reduces to $w(T) + \sum_{i \in ch(T)-L} \text{ALGO}(\mathbf{T}_i)$. Again, it follows from the recursive nature of the algorithm that:

$$\begin{aligned} \text{ALGO}(T) &= w(T) + \sum_{i \in ch(T)-L} \left[w(\mathbf{T}_i) + \sum_{j \in ch(\mathbf{T}_i)} \text{ALGO}(\mathbf{T}_j) \right] \\ &\leq 2w(T) + \sum_{i \in ch(T)-L} \sum_{j \in ch(\mathbf{T}_i)} \text{ALGO}(\mathbf{T}_j) \end{aligned}$$

Define $\mathbb{S}$ as the set of nodes of T reached in exactly two hops in $T + A$, namely $\mathbb{S} = \{j \in T : (r \to i \to j) \in T + A\}$. Noticing that all nodes in

$\{ch(\mathbf{T}_i)\}_{i \in ch(T)-L}$ also belong to $\mathbb{S}$, the last inequality reduces to:

$$\text{ALGO}(T) \leq 2w(T) + \sum_{i \in \mathbb{S}} \text{ALGO}(\mathbf{T}_i) \tag{3}$$

We argue that for any $i \in \mathbb{S}$ which is an internal node of T, $w(\mathbf{T}_i) \leq w(T)/(k+1)$. This result relies on the following observation about the auxiliary weight function w' that can be readily verified: for every subtree T' of T such that $r(T')$ is not a leaf of T, $w(T') \leq w'(T')$.

So consider a node $j \in \mathbb{S}$ which is an internal node of T, and let $(r \rightarrow i \rightarrow j)$ be the path from r to j in $T + A$. Clearly this implies that i is also an internal node of T. Suppose that i is not an endpoint of a hotlink in A; this implies that i is a child of r in T. Moreover, i belongs to U^p, otherwise it would be the root of a tree U^x with $x \neq p$ and consequently be pointed by a hotlink from r. Therefore $\mathbf{T}_i = U^p_i$ and by the definition of U^p_i it follows that $w(\mathbf{T}_i) \leq w'(\mathbf{T}_i) = w'(U^p_i) \leq w(T)/(k+1)$. Notice that j cannot belong to any tree U^x with $x \neq p$, otherwise i would not be in the path from r to j in $T + A$. Therefore $j \in U^p$ and, because j must be a descendant of i, $j \in U^p_i$. Then it is not difficult to see that the execution of the algorithm over $U^p_i = \mathbf{T}_i$ must be the one that calls the execution of the algorithm over $\mathbf{T}_j$, and therefore $\mathbf{T}_j \subseteq \mathbf{T}_i$. As a consequence, we have that $w(\mathbf{T}_j) \leq w(T)/(k+1)$ and the claim holds for this case.

Now suppose that i is the endpoint of a hotlink in A, that is, a hotlink from r. It follows that $i = r(U^x)$ for some $x \neq p$. In addition, j cannot belong to a tree U^z different than U^x, otherwise the path form r to j in $T + A$ would not be $(r \rightarrow i \rightarrow j)$. Using the same reasoning as in the previous case, this implies that $\mathbf{T}_j \subseteq U^x$ and consequently $\mathbf{T}_j \subseteq U^x_j$. As j is not a leaf of T, we have that $w(\mathbf{T}_j) \leq w(U^x_j) \leq w'(U^x_j) \leq w(T)/(k+1)$ and the claim follows.

Therefore, defining $\mathbb{S}'$ as the set of nodes in $\mathbb{S}$ which are not leaves of T and noticing again that for any leaf i of T we have $\text{ALGO}(\mathbf{T}_i) = 0$, inequality (3) reduces to

$$\text{ALGO}(T) \;\leq\; \sum_{i \in \mathbb{S}'} \text{ALGO}(\mathbf{T}_i) + 2w(T) \tag{4}$$

where $w(\mathbf{T}_i) \leq w(T)/(k+1)$ for every $i \in \mathbb{S}'$.

For each subtree $T' \subseteq T$ we define $w|T'$ as the weights w restricted to the nodes of T', or alternatively $(w|T')(j) = w(j)$ for all $j \in T'$ and $(w|T')(j) = 0$ for all $j \notin T'$. In order to simplify the notation, we can see every weight function as a multiset of nonnegative numbers.

Now we proceed by induction. Suppose that for any proper subtree T' of

T ALGO$(T') \leq \frac{2H(w|T')}{\log(k+1)} + 2w(T')$. This expression clearly holds if T' is only a leaf.

The following lemma is a slightly generalization of the one proved in [KKS01a]:

Lemma 24 *Let w be a weight multiset and $\{w^1, \ldots, w^a\}$ disjoint submultisets of w. Let $W = \sum_j w(j)$ and $W^i = \sum_j w^i(j)$. Then:*

$$H(w) \geq \sum_{i=1}^{a} H(w^i) + \sum_{i=1}^{a} W^i \log \frac{W}{W^i}$$

Proof:

$$\begin{aligned} H(w) &= \sum_j w(j) \log \frac{W}{w(j)} \geq \sum_{i=1}^{a} \sum_j w^i(j) \log \frac{W}{w^i(j)} \\ &= \sum_{i=1}^{a} \sum_j w^i(j) \log \left(\frac{W^i}{w^i(j)} \cdot \frac{W}{W^i} \right) \\ &= \sum_{i=1}^{a} \sum_j w^i(j) \log \frac{W^i}{w^i(j)} + \sum_{i=1}^{a} W^i \log \frac{W}{W^i} \\ &= \sum_{i=1}^{a} H(w^i) + \sum_{i=1}^{a} W^i \log \frac{W}{W^i} \end{aligned}$$

■

Defining $a = 2/\log(k+1)$ and employing the inductive hypothesis to inequality (4) we have:

$$\begin{aligned} \text{ALGO}(T) &\leq a \sum_{i \in \mathbb{S}'} H(w|\mathbf{T}_i) + 2 \sum_{i \in \mathbb{S}'} w(\mathbf{T}_i) + 2w(T) \\ &\leq aH(w) - a \sum_{i \in \mathbb{S}'} w(\mathbf{T}_i) \log \frac{w(T)}{w(\mathbf{T}_i)} + 2 \sum_{i \in \mathbb{S}'} w(\mathbf{T}_i) + 2w(T) \\ &\leq aH(w) - a \sum_{i \in \mathbb{S}'} w(\mathbf{T}_i) \log(k+1) + 2 \sum_{i \in \mathbb{S}'} w(\mathbf{T}_i) + 2w(T) = aH(w) + 2w(T) \end{aligned}$$

where the second inequality follows from Lemma 24 and the third inequality follows from the fact that $w(\mathbf{T}_i) \leq w(T)/(k+1)$ for all $i \in \mathbb{S}'$. This completes the inductive step and proves that the cost of the assignment returned by the algorithm can be upper bounded by $\frac{2H(w)}{\log(k+1)} + 2w(T)$.

Algorithm for k-HAP with internal weights

Let T be a tree rooted at node r and w a weight function which might be positive for internal nodes of T. Define $\overline{w}(r) = 0$ and $\overline{w}(u) = w(u)$ for

every $u \in T$ different than r. Clearly $\mathrm{EP}(T + A, w) = \mathrm{EP}(T + A, \overline{w})$ for all assignments A. In addition, we note that the optimal solution for $(T, \overline{w})$ costs at least $\overline{w}(T)$.

Now we create an auxiliary tree T' from T by adding one leaf l_u to each node u of T and define $w'(l_u) = \overline{w}(u)$ and $w'(u) = 0$ for all $u \in T$. Clearly each assignment for T is a no-leaf assignment for T' and vice-versa. So consider a no-leaf assignment A' for T'. Then the expected user path in $T' + A'$ is:

$$\begin{aligned}
\mathrm{EP}(T', A', w') &= \sum_{l_u \in T'} d(r, l_u, T' + A') w'(l_u) = \sum_{l_u \in T'} (d(r, u, T' + A') + 1) w'(l_u) \\
&= \sum_{u \in T} d(r, u, T' + A') \overline{w}(u) + \overline{w}(T) = \sum_{u \in T} d(r, u, T + A') \overline{w}(u) + \overline{w}(T) \\
&= \mathrm{EP}(T, A', \overline{w}) + \overline{w}(T)
\end{aligned}$$

where the forth inequality follows from Proposition 1.

Using this relationship between $\mathrm{EP}(T', A', w')$ and $\mathrm{EP}(T, A', \overline{w})$, it is easy to see that if A^* is an optimal no-leaf assignment for T' then it is also an optimal assignment for T. Moreover, if A' is an α-approximate no-leaf assignment for T' then it is a 2α-approximate assignment for T:

$$\begin{aligned}
&\mathrm{EP}(T, A', \overline{w}) \leq \mathrm{EP}(T', A', w') \leq \alpha \mathrm{EP}(T', A^*, w') \\
&= \alpha(\mathrm{EP}(T, A^*, \overline{w}) + \overline{w}(T)) \leq \alpha 2 \mathrm{EP}(T, A^*, \overline{w})
\end{aligned}$$

where the last inequality holds because the optimal solution for $(T, \overline{w})$ costs at least $\overline{w}(T)$. Combining with the fact that $\mathrm{EP}(T, A, w) = \mathrm{EP}(T, A, \overline{w})$ for all assignments A, it follows that A' is a 2α-approximation for the original instance (T, w).

Therefore, we can use the algorithm presented in the previous section to find a $\frac{4H(w)}{\log(k+1)} + 4w(T)$ approximation for the k-HAP with arbitrary non-negative weights.

(b) Proof of Lemma 10

Consider a node $u \in \overline{T}_q$. Let (c_q, x) be a hotlink or an arc in $T + A^*$ such that $x \in \overline{T}_q$. By means of contradiction, assume that c_q does not belong to the path $P = (r \rightsquigarrow u)$ in $T + A^*$. Then Proposition 4 implies that $P = (r \rightsquigarrow s \rightarrow s' \rightsquigarrow u)$ where s and s' are respectively a proper ancestor and a proper descendant of c_q and (s, s') is a hotlink in A^*. Notice that s belongs to Q. Now we have two cases: if s' is a descendant of x, then $s' \in \overline{T}_q$ and the fact that s is a proper ancestor of c_q contradicts the definition of the latter. On the other hand, if s' is a 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. This concludes the proof of the lemma.

B
Binary Searching in Trees

B.1 Searching in path-like trees

In this section, we argue that the problem of searching in path-like trees can be reduced to the well-known problem of searching in ordered lists with different access probabilities. The latter problem is defined as follows. As input we have a sequence $L = \{l_1, \ldots, l_n\}$ of (real) numbers in increasing order and a probability distribution $P = \{p_1, \ldots, p_n; q_0, \ldots, q_n\}$. One number, which may not belong to L, is marked; p_i is the probability that l_i is the marked number and q_i is the probability that the marked number is in the interval $I_i = (l_i, l_{i+1})$ (using the convention that $l_0 = -\infty$ and $l_{n+1} = \infty$). One may query an element l_i of L and receive the information whether the marked node is greater than l_i, less than l_i or is l_i itself.

As in the case of searching in trees, every search strategy for this problem can be represented by means of a binary search tree D [PS93]. Such binary tree contains n internal nodes, each corresponding to a different element of L, and $n + 1$ leaves, each corresponding to an interval I_i. In addition, each internal node u of D satisfies the search property, namely if u corresponds to the number l_i then: (i) all nodes of the right subtree of u correspond to either a query for a number l_j with $j > i$ or to an interval I_j with $j \geq i$; (ii) all nodes of the left subtree of u correspond to either a query for a number l_j or to an interval I_j, both with $j < i$. If $d(l_i)$ is the distance from $r(D)$ to the node of D which corresponds to l_i and $d(I_i)$ is the distance from $r(D)$ to the node of D which corresponds to I_i, then the cost of D is given by $\text{cost}(D, P) = \sum_{i=1}^{n} d(l_i)p_i + \sum_{i=0}^{n} d(I_i)q_i$.

Then the problem of searching in ordered lists with different access probabilities (SOL) can be stated as follows: given an instance (L, P), the objective is to find a decision tree for (L, P) with lowest cost.

Now we start presenting the relation between searching in ordered lists and searching in path-like trees. Consider an instance (T, w) of the latter problem, where $T = \{t_1, \ldots, t_n\}$ is a path rooted at t_1. We create the

following SOL instance: L consists of the first $n-1$ natural numbers and $P = \{p_1, \ldots, p_{n-1}; q_0, \ldots, q_{n-1}\} = \{0, \ldots, 0; w(t_1), \ldots, w(t_n)\}$. The idea is that (T, w) is associated with (L, P) via the function ϕ, where for each arc (t_i, t_{i+1}) of T we have $\phi((t_i, t_{i+1})) = i$ and for each node t_i of T we have $\phi(t_i) = I_{i-1}$.

Consider a decision tree D for (T, w). Using the association given by ϕ we can create the following decision tree D' for (L, P): D' is exactly the tree D but with different 'correspondences'; if a node u of D corresponds to an arc (i, j) (or node t_i) of T, then the node u of D' corresponds to the number $\phi((i, j))$ (or to the interval $\phi(t_i)$). In order to prove that D' is indeed a decision tree for (L, P) it suffices to show that its nodes satisfy the search property.

Consider an internal node u of D' which corresponds to a number $\phi(t_i) = i$. In addition, consider an internal node v of the right subtree of u in D' which corresponds to a number $\phi(t_j) = j$. Due to the construction of D', the node u of D corresponds to (t_i) and the node v of D corresponds to (t_j). In addition, v belongs to the right subtree of u in D and hence $t_j \in T_{t_i}$, or alternatively, $j > i$. Now consider a leaf v of the right subtree of u in D' which corresponds to the interval $\phi((t_j, t_{j+1}))$. Using the same arguments, it is easy to see that (t_j, t_{j+1}) is an arc of T_{t_i}, that is, $j \geq i$. It then follows from the fact that $l_i = i$ for all $1 \leq i \leq n-1$ that u in D' satisfies condition (i) of the search property. An analogous result can be easily proved for the left subtree of u, which implies that D' is a decision tree for (L, P).

Moreover, the construction of the probability distribution P ensures that the cost of D' is the same as the cost of D.

The reverse association also holds: for each decision tree D for (L, P), we can find a decision tree D' for (T, w) such that both have the same cost. This result follows by by applying the same reasoning as done previously.

Due to these associations, it is not difficult to see that an approximate algorithm for the SOL problem gives an approximation for searching in path-like trees. In effect, it follows that if D^* is an optimal decision tree for (L, P) then the associated decision tree $D^{*\prime}$ is also optimal for (T, w). Therefore, if an algorithm for SOL finds a decision tree D such that $\text{cost}(D, P) \leq \alpha \cdot \text{cost}(D^*, P)$, it follows that the associated decision tree D' for (T, w) has cost $\text{cost}(D', w) = \text{cost}(D, P) \leq \alpha \cdot \text{cost}(D^*, P) = \alpha \cdot \text{cost}(D^{*\prime}, w) = \alpha \text{OPT}(T, w)$.

B.2 Proofs of lemmas

Given any tree T' and two nodes $u, v \in T'$, a lowest common ancestor of u and v is a node $x \in T'$ 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 5 *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 12 *Consider a tree T and a decision tree D for T. For each subtree T' of T, there is a unique node $u \in D$ which is the representative of T' in D.*

Proof: Consider a subtree T' of T. We define $u(T')$ as the node of D closest to $r(D)$ which corresponds to an arc or to a node of T'. More formally, $u(T') = u_\alpha$ such that $\alpha = \text{argmin}\{d(r(D), u_{\alpha'}, D) : \alpha' \text{ is a node or an arc of } T'\}$. We claim that $u(T')$ is a representative of T', and to prove this it suffices to argue that $u(T')$ is an ancestor of all nodes of D that corresponds to nodes or arcs of T'.

By means of contradiction, let β be a node or an arc of T' such that u_β is not a descendant of $u(T') = u_\alpha$. The node u_β cannot be a proper ancestor of u_α, or that would contradict the choice of u_α. Thus, from Proposition 5 u_α and u_β are on subtrees rooted at different children of their lowest common ancestor, say $u_{(i,j)}$. Without loss of generality, assume that u_β is belongs to the right subtree of $u_{(i,j)}$ and that u_α belongs to the left subtree of $u_{(i,j)}$. By definition, β is an arc/node of T_j and α is an arc/node of $T - T_j$. Thus, the root of T' must be a proper ancestor of j so that both α and β belong to T'. But this implies that $(i, j) \in T'$. Because $u_{(i,j)}$ is a proper ancestor of u_α, it is closer to the root of D and contradicts the choice of u_α. Therefore, $u(T')$ must be an ancestor of all nodes of D that corresponds to arcs/nodes of T'.

Notice that $u(T')$ is the unique representative of T', otherwise a different representative would have to be a proper ancestor of $u(T')$ and would contradict the fact $u(T')$ is a representative. ■

Lemma 13 *Consider a tree T, a weight function w and a decision tree D for T. Then for every subtree T' of T, $\sum_{v \in T'} d(u(T'), u_v, D)w(v) \geq OPT(T', w)$.*

Proof: Consider a subtree T' of T. The idea is to construct a decision tree D' for T' by removing all nodes of D which do not correspond to elements of T', as illustrated in Figure 7.1. For that we define the recursive function $rec(u_\alpha)$ which receives a node u_α of D and outputs a tree in the following way: if α is a node of T', return u_α; if α is an arc of T', compute the trees $rec(r)$ and $rec(l)$ for the right and left child of u_α (respectively r and l) and return the tree formed by u_α with $rec(r)$ appended as its right child and $rec(l)$ appended as its left child; if α is not a node or an arc of T' then return $rec(r)$ if α is an ancestor of the root of T' or return $rec(l)$ otherwise. The tree D' is then defined as $D' = rec(u(T'))$.

Claim 1: D' contains a node corresponding to each arc/node of T'. Consider an arc (i,j) of T. Notice that if $(i,j) \notin T'$, only nodes in the right subtree of $u_{(i,j)}$ can correspond to elements of T' (in case j is an ancestor of $r(T')$) or only nodes in the left subtree of $u_{(i,j)}$ can correspond to elements of T' (in case j is not an ancestor of $r(T')$). Using this observation, it follows by induction on the subtrees of D that the nodes of $rec(u(T'))$ are exactly the nodes of $D_{u(T')}$ which correspond to arcs/nodes of T'. The claim then follows from Lemma 12.

Claim 2: If α is a node (arc) of T', then u_α is a leaf (internal node) of D'. If α is a node of T', then u_α is a leaf of D and, due to the construction of D' and using Claim 1, u_α is also a leaf of D'. If $\alpha = (i,j) \in T'$, we notice that $D_{u_{(i,j)}}$ contains the leaf $u_j \neq u_{(i,j)}$. It follows from the fact that for each $u \in D'$, $D'_u \subseteq D_u$ and from Claim 1 that $D'_{(i,j)}$ contains the node $u_j \neq u_{(i,j)}$ and hence $u_{(i,j)}$ is an internal node of D'.

Because for each $u \in D'$, D'_u contains exactly the nodes of D_u which correspond to arcs/nodes of T', along with the fact that each internal node of D satisfies the search property, each internal node of D' also satisfies the search property. Together with the previous claims, we have that D' is a valid decision tree for T'.

Finally, it is easy to see (and to prove by induction) that for any two nodes x and y which belong to both D and D', we have $d(x,y,D') \leq d(x,y,D)$. Recalling that $u(T')$ is the root of D', we have that $\text{cost}(D',w) \leq \sum_{v \in T'} d(u(T'),u_v,D)w(v)$. As D' is a valid decision tree for T', we have that $\text{OPT}(T',w) \leq \text{cost}(D',w)$ and the result follows. ■

Lemma 14 $\sum_{q_i \in Q} d(r^*, u(\overline{T}_{q_i}), D^*) w(\overline{T}_{q_i}) \geq H(\{w(\overline{T}_{q_i})\})/\log 3 - w(T)$

Proof: Construct the tree D' by adding new leaves to D^* as follows: 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 of D' have zero probability.

Clearly the distance from $r(D')$ to l_i in D' equals to $d(r^*, u(\overline{T}_{q_i}), D^*) + 1$. Because only the nodes $\{l_i\}$ have nonzero probability, the cost of D' 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 at most one new leaf $\{l_i\}$ is added to each node $\{u(\overline{T}_{q_i})\}$ and D' is at most a ternary tree. We can then use Shannon Coding Theorem [Gallager68] to bound the cost of D' as $\text{cost}(D',w') \geq$

$H(\{w'(l_i)\})/\log 3$. Substituting this bound in the previous displayed inequality and noticing that $H(\{w(\overline{T}_{q_i})\}) = H(\{w'(l_i)\}) \cdot w(T)$, we have:

$$\frac{H(\{w(\overline{T}_{q_i})\})}{\log 3} \leq \sum_{q_i} \left(d(r^*, u(\overline{T}_{q_i}), D^*) + 1\right) w(\overline{T}_{q_i})$$

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