Part I

k-Hotlink Assignment Problem

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 (in number of arcs) 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 positive weights on internal nodes can be modeled via 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 1.1).

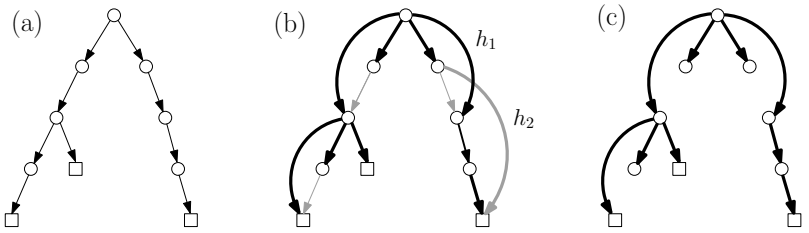

Figure 1.1: (a) Original tree. (b) Enhanced tree, with greedy paths in bold. Hotlinks h_1 and h_2 are crossing. (c) Tree induced by user paths.

Like most of the papers in this subject, here we assume the greedy user model. Henceforth, for every enhanced tree $T + A$ and for every pair of nodes $u, v \in T$, *the path* or *the user path* from u to v in $T + A$ both refer to the path that users under the greedy model follow when going from u to v in $T + A$.

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 [PE97]. In [CKK+03], 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 [PLS04, Jacobs08]. Turning our attention to theoretical results, in [BKK+00] 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 Δ 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 [KKS01a], 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 Δ). In [DL05] and [DL06], 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 [DL06], 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 [Jacobs07] – 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. [GKM+03] and Pessoa et. al. [PLS04] (see also [KGL+07] for a journal version merging both papers). The algorithm of [GKM+03] 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 1.1.c). We denote such tree by T^A. The algorithm of [PLS04], which can be viewed as an optimized version of the one proposed in [GKM+03], has the following property: for each integer D, it calculates in $O(n3^D)$ 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 [BKL+03, MP04, PLS04b]. In [BKL+03], Bose et. al. discuss the use of hotlink assignments in asymmetric communication protocols [AM01] 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 with respect to the maximum gain for DAG's [MP04]. In [Jacobs07], 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 [PLS04], 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 fairly 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 4.

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 [DL05] 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 [BKK+00] – 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 5.

We shall notice that the complexity of both the 1-HAP and k-HAP remains open. In addition, it is worth mentioning that the complexity of these problems contrasts with their 'worst case' versions that are polynomially solvable [PLS04b]. 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 [Karp61, GKY02]. Finally, we remark that the problem of binary searching in trees presented in the next part of this work also shares this characteristic.