Consiglio Nazionale delle Ricerche

Efficient Strategies for Partitioning and Querying a Hierarchical Document Space

B. Codenotti, G. De Marco, M. Leoncini, M. Montangero, M. Santini

IIT TR-25/2002

Technical report

Dicembre 2002

Istituto di Informatica e Telematica

Efficient Strategies for Partitioning and Querying a Hierarchical Document Space

Bruno Codenotti* Gianluca De Marco† Mauro Leoncini‡ Manuela Montangero§
Massimo Santini¶

Abstract

We consider a problem arising in the efficient management of a Hierarchical Document Space, i.e., partitioning the leaves of a tree among a set of servers in a such a way that it is possible to take full advantage of the hierarchical system to efficiently answer user's queries. After proving that the problem is NP-Hard, we devise efficient approximate solutions, and we make a number of experiments which show that allowing for very little space inefficiency can be instrumental to achieving a significant improvement in the query efficiency.

Keywords: Algorithm analysis, information storage and retrieval, approximation algorithms, hierarchical document collections.

1 Introduction

A wide body of work in Information Retrieval (IR) addresses issues such as creating and maintaining hierarchically structured collections of documents. The exponential growth of the web, with the related difficulties of finding and organizing relevant information, has done nothing but increase the interest, at the scientific as well as commercial level, towards the development of efficient hierarchical retrieval systems. Indeed hierarchical web retrieval systems offer a number of potential advantages over flat keyword based search utilities. As an example, they can offer greater query precision and much faster response times [1, 7].

Roughly speaking, the different approaches adopted to produce structured ontologies can be divided into two categories, namely classification and clustering systems. Successful hierarchical topic taxonomies have been manually created before the computing era (e.g., the Dewey Decimal Classification system, published in 1876 for the first time) and of course recently, in connection with the development of the web (e.g., the Yahoo! directory). Automatic classification systems have also been proposed to overcome the difficulties of manually compiling very large directories [2, 1, 6]. However, automatic classification system have to confront a number of challenging problems, such as incrementality and classification reliability. These issues have been at least partially dealt with for moderate size data sets, but are far from being solved for the huge data sets typically associated with the web.

A complementary approach to the automatic construction of structured ontologies is given by clustering (see, e.g., [12]). With respect to classification, clustering has the advantage that the topics need not be defined in advance. Indeed, clustering is the problem of automatically assigning class labels to the documents. Clustering is a widely investigated problems in many fields (such as Machine Learning, Data Mining, Computational Geometry, and of course Information Retrieval). However, there is no clear

*Dept. of Comp. Sci.,The Univ. of Chicago, On leave from IIT – CNR, e-mail codenott@cs.uchicago.edu

†Ist. di Informatica e Telematica, Via G. Moruzzi 1, I-56100 Pisa (Italy), e-mail demarco@iit.cnr.it

‡Dip.to di Scienze Sociali, Cognitive e Quantitative, Via Giglioli Valle 9, I-42100 Reggio Emilia (Italy), e-mail leoncini@unimo.it

§Ist. di Informatica e Telematica, Via G. Moruzzi 1, I-56100 Pisa (Italy), e-mail montangero@iit.cnr.it

¶Dip.to di Scienze Sociali, Cognitive e Quantitative, Via Giglioli Valle 9, I-42100 Reggio Emilia (Italy), e-mail msantini@unimo.it

indication as to whether or not existing algorithms could effectively be employed in large scale web applications (see, e.g., [4, 5], for a discussion of the difficulties connected to the efficient clustering of very large document collections and for sequential and distributed algorithms with "state of the art" performances).

In this paper we isolate a problem, which we call *Minimum Redirections Problem*, related to the maximum efficiency achievable when querying a hierarchically structured corpus, and look at it from an algorithmic viewpoint.

More precisely, the Minimum Redirection Problem consists of partitioning the set of leaves of a tree among a set of servers in such a way that the number of servers that store the leaves of any subtree is minimum (on the average) over the set of possible partitions. Formally, let $T = (V, E)$ be a (rooted) tree and, for any node v of T, let $L(v)$ denote the set of leaves of the subtree rooted at v. We want to find a partition $\mathcal{P}$ of the leaves of T such that the quantity

$$\sum_{v \in V} |\{A \in \mathcal{P} \mid L(v) \cap A \neq \emptyset\}| \tag{1}$$

is minimum over the set of all the partitions. Note that, for a given partition $\mathcal{P}$, measure (1) over $|\mathcal{P}|$ represents the average number of sets from $\mathcal{P}$ having non empty intersection with the descendant of any internal node v.

Assume that the internal nodes and the leaves of the tree represent topics (categories) and documents, respectively. Then a query issued at a given internal node v will be redirected to those servers that store the documents associated with the subtree rooted at v. Within this framework, minimizing (1) amounts to minimizing the average number of servers under the assumption that queries are issued against any two different categories with the same probability. Since we have no available data on the frequency distribution of topic search (or direct browsing), we regard (1) as a rough measure that applies to a worst case scenario.

The outline of the paper is the following. In Section 2 we define the Minimum Redirection Problem formally and prove that it is an NP-hard optimization problem by a reduction from Minimum Bin Packing. We then prove an inapproximability result for a measure closely related to (1), for which the reduction is approximation preserving. We address the problem of approximating optimal redirections in Section 3. We give an efficient 2-approximation algorithm for Minimum Redirection Problem, which uses a subroutine to approximately solve certain Minimum Bin Packing instances, and prove that the approximation ratio is tight by showing a family of trees for which the output is arbitrarily close to twice the optimal value. We compare our algorithm with a simple algorithm which minimizes the number of sets in the partition (a measure related to the number of servers needed to store the document collection). Although for some family of trees such simple algorithm computes also a provably better approximation to (1), we experimentally observe that, in general, space efficiency comes at the expenses of a reduced query redirection performance. The section on the experimental results (Section 4) describes other outcomes of running our algorithm (with two different Bin Packing subroutines) on three different data sets: *Open Directory Project* (ODP), *Yahoo!*, and *Arianna* [13, 14, 15]. The most promising evidence suggested by the results is that allowing for a little extra space (servers) may make it possible to obtain big savings in query redirections.

2 Minimum Redirection Problem

In this section we formally introduce the Minimum Redirection Problem and we show that it is NP-hard. The following notation will be used for trees: given a tree T and a vertex v of the tree, $S_T(v)$ will denote the subtree rooted at v, $L_T(v)$ and $N_T(v)$ will denote the set of leaves and internal nodes of $S_T(v)$, respectively. We will drop the subscript when T is clear from the context, and we will omit to specify v to intend the root, i.e., L are the leaves of T.

Definition 1 *(Number of redirections) Let $T = (V, E)$ be a tree and $\mathcal{P}$ a partition of L_T. The* number of redirections *for a node v of T given $\mathcal{P}$ is*

$$\rho_T(v, \mathcal{P}) = |\{A \in \mathcal{P} \mid L_T(v) \cap A \neq \emptyset\}|;$$

similarly, the total *number of redirections for the tree T given $\mathcal{P}$ is*

$$\rho_T(\mathcal{P}) = \sum_{v \in N_T} \rho_T(v, \mathcal{P}),$$

and the average *number of redirections for the tree T given $\mathcal{P}$ is*

$$\bar{\rho}_T(\mathcal{P}) = \rho_T(v, \mathcal{P})/|N_T|.$$

Definition 2 *(Number of extra redirections) Let $T = (V, E)$ be a tree and $\mathcal{P}$ a partition of L_T. The number of* extra *redirections for a node v of T given $\mathcal{P}$ is*

$$\chi_T(v, \mathcal{P}) = \rho_T(v, \mathcal{P}) - 1.$$

Similarly, $\chi_T(\mathcal{P}) = \sum \chi_T(v, \mathcal{P})$ and $\bar{\chi}_T(\mathcal{P}) = \chi_T(v, \mathcal{P})/|N_T|$.

Given vertex v, the number $\rho_T(v, \mathcal{P})$ counts how many sets of partition $\mathcal{P}$ contain leaves of subtree $S_T(v)$, and can be regarded as the number of servers to which a query against v will be redirected. The corresponding measure $\chi_T(v, \mathcal{P})$ represents the number of extra redirections, considering that the query must be sent to at least one server. The corresponding average values are defined under the hypothesis that queries are issued against any two different categories with the same probability.

Definition 3 *(Minimum Redirection Problem) Given a tree T and a positive integer* capacity *c, find a partition $\mathcal{P}$ of L_T such that*

1. *for every set $A \in \mathcal{P}$ the cardinality of A is bounded by c;*
2. *the total number of redirections $\rho_T(\mathcal{P})$ is minimized.*

A feasible solution *is a partition $\mathcal{P}$ satisfying only condition 1.*

Lemma 1 *Given a tree $T = (V, E)$ and a feasible solution $\mathcal{P}$ for capacity c, for every vertex $v \in V$,*

$$\frac{|L_T(v)|}{c} \leq \rho_T(v, \mathcal{P}) \leq \sum \rho_T(u, \mathcal{P}),$$

where the summation extends over the children of v.

Proof. A set of the partition can not contain more than c leaves, thus $|L_T(v)|/c$ sets are necessary. The union of the partitions given by the children of v is a feasible solution. ◇

We show that the Minimum Redirection Problem is NP-hard by reduction from Minimum Bin Packing.

Definition 4 *(Minimum Bin Packing) Given a finite set of items with positive integer sizes $s_1, \ldots, s_n$ and a positive integer capacity c, find a partition $\mathcal{Q}$ of $\{1, \ldots, n\}$ such that $\sum_{i \in A} s_i \leq c$ for every $A \in \mathcal{Q}$ and $|\mathcal{Q}|$ is minimum. We shall refer to $\sum_{i \in A} s_i$ as to the* level *of A.*

The Minimum Bin Packing problem is NP-hard, approximable within 3/2 [11] but not approximable within $3/2 - \epsilon$ for any $\epsilon > 0$ [8, 3]. Nevertheless it admits a FPTAS$^\infty$ and is approximable within $1 + \epsilon$ in time polynomial in $1/\epsilon$, where $\epsilon = O(\log^2(opt)/opt)$ [10].

First of all, we prove a simple lemma that we will use in the subsequent hardness and inapproximability theorems.

Lemma 2 *For every tree T having all the leaves at distance 2 from the root and every partition $\mathcal{P}$ of the leaves, there exists a partition $\mathcal{P}'$ such that*

1. *$\rho_T(\mathcal{P}') \leq \rho_T(\mathcal{P})$,*

2. *for every v different from the root, $L_T(v) \subseteq A' \in \mathcal{P}'$.*

Proof. We iterate the following argument over all $v \in V$ such that $\rho_T(v, \mathcal{P}) > 1$. Let $\hat{v} \in V$ be one such node.

Define $\mathcal{P}' = \{A' \neq \emptyset \mid A' = A \setminus L_T(\hat{v}), A \in \mathcal{P}\} \cup L_T(\hat{v})$. Then, $\rho_T(\hat{v}, \mathcal{P}') = 1 < \rho_T(\hat{v}, \mathcal{P})$. If v is not the root and $v \neq \hat{v}$, we have $\rho_T(v, \mathcal{P}') = \rho_T(v, \mathcal{P})$. Finally, if r is the root, $\rho_T(r, \mathcal{P}') = |\mathcal{P}'| \leq |\mathcal{P}| + 1 = \rho_T(r, \mathcal{P}) + 1$.

Hence $\rho_T(\mathcal{P}') \leq \rho_T(\mathcal{P})$. ◇

Theorem 1 *The Minimum Redirection Problem is NP-hard.*

Proof. Reduction is from Minimum Bin Packing. Suppose we are given an instance of Minimum Bin Packing with capacity c and items of size $s_1, \ldots, s_n$. We define tree T as in Figure 1.

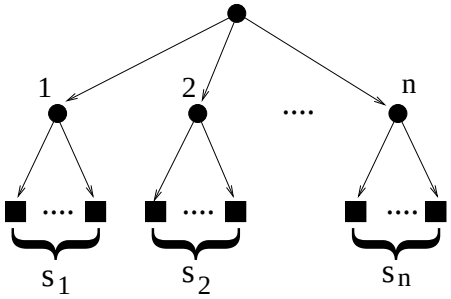

Figure 1: Tree T constructed in the reduction from Min Bin Packing on input $(c; s_1, \ldots, s_n)$.

Let $\mathcal{P}$ be a feasible solution to the Minimum Redirection Problem on T with capacity c. By Lemma 2, we can assume that $\rho_T(i, \mathcal{P}) = 1$ for every $1 \leq i \leq n$.

The partition $\mathcal{Q}$ of $\{1, \ldots, n\}$ given by the sets $\{i \mid L_T(i) \subseteq A \in \mathcal{P}\}$, for $1 \leq i \leq n$, has the same cardinality of $\mathcal{P}$ and gives immediately a feasible solution to the Minimum Bin Packing problem. Moreover, $\rho_T(\mathcal{P}) = |\mathcal{Q}| + n$.

Now, if $\mathcal{P}$ is such that $\rho_T(\mathcal{P})$ is minimum, then $|\mathcal{Q}|$ will also be minimum. ◇

The same hardness result clearly holds for the problem of minimizing $\chi_T(\mathcal{P})$. For the latter, the following stronger inapproximability result holds

Theorem 2 *The minimum number of extra redirection is not approximable within $3/2 - \epsilon$ for any $\epsilon > 0$.*

Proof. The same reduction as in Theorem 1 holds, with $\chi_T(\mathcal{P}) = |\mathcal{Q}| - 1$. The result then follows from the inapproximability of Minimum Bin Packing cited above. ◇

3 Approximation algorithm

In this section we present a recursive 2-approximation algorithm to solve the Minimum Redirection Problem. We show that the analysis of the approximation factor is tight by giving a family of trees for which the approximation factor approaches 2.

The input of the recursive call is a vertex v and the output is a partition of $L(v)$: if v is a leaf then the output is simply the singleton composed by v. Otherwise a recursive call is made on the children of v collecting the results in a partition $\mathcal{P}$ of $L(v)$. Considering the cardinality of each set in $\mathcal{P}$ as an item size, the algorithm computes a feasible solution to the Minimum Bin Packing problem. Finally, the recursive call returns a coarser partition obtained from $\mathcal{P}$ by merging the sets that have been packed together.

Let $\mathrm{BP}(c; s_1, \ldots, s_n)$ be an approximation algorithm for the Minimum Bin Packing problem. We say that BP is a *reasonable bin packer* if the levels $l_1 \geq l_2 \geq \ldots \geq l_m$ of its solution are such that

1. $l_i > c/2$, for all $1 \leq i < m$,

2. $l_{m-1} + l_m > c$.

These conditions together simply say that the solution is minimal, in the sense that any two sets in the partition can not be merged together without violating the capacity constraint.

The recursive procedure of the approximation algorithm for the Minimum Redirection Problem is the following:

`Approx-MRN`(v)
if v is a leaf then return $\{v\}$;
for every child u of v, $\mathcal{P}_u =$ `Approx-MRN`(u);
$\{P_1, \ldots, P_n\} = \bigcup_u \mathcal{P}_u$;
$\mathcal{Q} = \mathrm{BP}(c; |P_1|, \ldots, |P_n|)$;
$\mathcal{P} = \{A | A = \bigcup_{i \in Q} P_i, Q \in \mathcal{Q}\}$;
return $\mathcal{P}$.

Let r be the root of tree T, then $\mathcal{P} =$`Approx-MRN`(r). is the solution of the Minimum Redirection Problem.

We now prove that `Approx-MRN` is a 2-approximation algorithm.

Theorem 3 *Given a tree T and a positive integer capacity c, let $\mathcal{P}_{opt}$ be the optimal partition of the leaves and $\mathcal{P}_A$ the one found by* ***Approx-MRN*** *using a reasonable bin packer. For every vertex v in tree T we have*

$$2 \cdot \rho_T(v, \mathcal{P}_{opt}) > \rho_T(v, \mathcal{P}_A).$$

Proof. Suppose that $\rho_T(v, \mathcal{P}_{opt}) < \rho_T(v, \mathcal{P}_A) = m$ and let $l_1 \geq l_2 \geq \ldots \geq l_m$ be the levels of the solution returned by `Approx-MRN`(v).

Since BP is reasonable, and observing that the sum of the levels is $|L_T(v)|$, we have

$$|L_T(v)| = \sum_{i=1}^{m-2} l_i + (l_{m-1} + l_m) > (m-2)c/2 + c = mc/2,$$

that implies

$$2 \cdot \frac{|L_T(v)|}{c} > m.$$

From Lemma 1 we have that $\rho_T(v, \mathcal{P}_{opt}) \geq |L_T(v)|/c$ and hence

$$2 \cdot \rho_T(v, \mathcal{P}_{opt}) \geq 2 \cdot \frac{|L_T(v)|}{c} > m = \rho_T(v, \mathcal{P}_A).$$

◇

Corollary 1 ***Approx-MRN*** *is a 2-approximation algorithm.*

Proof. Using the same notation as in Theorem 3, we have

$$\begin{aligned} 2 \cdot \rho_T(\mathcal{P}_{opt}) &= 2 \cdot \sum_v \rho_T(v, \mathcal{P}_{opt}) \\ &\geq \sum_v \rho_T(v, \mathcal{P}_A) \\ &= \rho_T(\mathcal{P}_A). \end{aligned}$$

◇

The simple analysis of Theorem 3 turns out to be accurate. In fact we can prove the following:

Theorem 4 *There exists a family of trees for which the performance ratio of* ***Approx-MRN*** *approaches 2.*

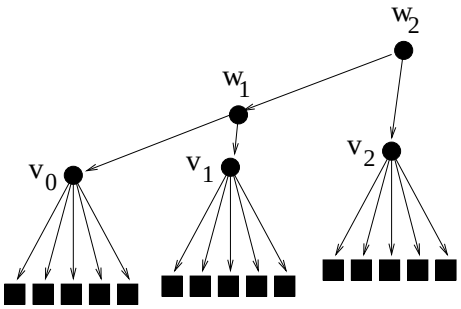

Figure 2: Example of T for $n = 2$ (and $c = 8$).

Proof. Given a positive integer $n > 0$ consider the pair (T, c) where $c = 4n$ and the tree T is defined as folows: the nodes are $\{l_k \mid 0 \leq k \leq (2n-1)(2n+1)\} \cup \{v_k \mid 0 \leq k < 2n-1\} \cup \{w_k \mid 1 < k < 2n-1\}$ and the edges are such that the l_k's are the leaves, the v_k's are such that $L(v_k) = \{l_i \mid \lfloor i/(2n+1) \rfloor = k\}$ (for $0 \leq k < 2n-1$), and, finally, the children of the w_k's are $\{w_{k-1}, v_k\}$ (for $1 < k < 2n-1$), while the children of w_1 are $\{v_0, v_1\}$.

First of all, we consider the solution $\mathcal{P}_A$ of `Approx-MRN` on input (T, c). As it is easy to check

$$\mathcal{P}_A = \{L(v_k) \mid 0 \leq k < 2n-1\}$$

since `Approx-MRN`$(v_k) = L(v_k)$ (for $0 \leq k < 2n-1$) and no such sets can be merged, since $|L(v_k)| = 2n+1 > c/2$. Hence it is straightforward to conclude that $\rho(v_k, \mathcal{P}_A) = 1$ (for $0 \leq k < 2n-1$). By construction,

$$L(w_k) = \bigcup_{0 \leq i < k+1} L(v_i), \tag{2}$$

so that $\rho(w_k, \mathcal{P}_A) = k+1$ (for $1 \leq k < 2n-1$), and hence

$$\rho(\mathcal{P}_A) = (2n-1) \times 1 + \sum_{1 \leq k < 2n-1} (k+1) = 2n^2 + O(n)$$

Now we consider the solution of an algorithm (called `Simple` from now onwards) that packs the leaves traversing the tree frontier from left to right in sets of size c. More formally, let $S_k = \{l_i \mid \lfloor i/4n \rfloor = k\}$, for $0 \leq k < \lceil |L|/4n \rceil = \lceil (4n^2-1)/4n \rceil = n$, then the solution $\mathcal{P}_S$ of `Simple` is

$$\mathcal{P}_S = \{S_k \mid 0 \leq k < n\}.$$

By straightforward inequalities, it is easy to check that

$$\begin{array}{ll} L(v_{2k}) \subseteq S_k & 0 \leq k < n \\ L(v_{2k+1}) \subseteq S_k \cup S_{k+1} & 0 \leq k < n-1 \end{array}$$

so that $\rho(v_k, \mathcal{P}_S) = 1 + (k \bmod 2) \leq 2$ (for $0 \leq k < 2n-1$); moreover, using also equation (2), one can conclude that (for $1 \leq k < 2n-1$)

$$L(w_k) \subseteq \bigcup_{0 \leq i < \lceil 1+k/2 \rceil} S_i$$

so that $\rho(w_k, \mathcal{P}_S) \leq \lceil 1 + k/2 \rceil$ (for $1 \leq k < 2n-1$). Hence

$$\rho(\mathcal{P}_S) \leq (2n-1) \times 2 + \sum_{1 \leq k < 2n-1} \lceil 1 + k/2 \rceil = n^2 + O(n)$$

by observing that

$$\begin{aligned}\sum_{1\le k<2n-1}\lceil 1+k/2\rceil &= \sum_{1\le i<n}\lceil 1+(2i)/2\rceil + \sum_{0\le i<n-1}\lceil 1+(2i+1)/2\rceil \\ &= \sum_{1\le i<n}(i+1) + \sum_{0\le i<n-1}(i+2) \\ &= 2\sum_{2\le i\le n} i.\end{aligned}$$

We can finally obtain the (asymptotic) approximation ratio of `Approx-MRN` on input (T, c). If if $\mathcal{P}_{opt}$ is the optimal solution, we have $\rho(\mathcal{P}_{opt}) \le \rho(\mathcal{P}_S)$ so

$$\frac{\rho(\mathcal{P}_A)}{\rho(\mathcal{P}_{opt})} \ge \frac{\rho(\mathcal{P}_A)}{\rho(\mathcal{P}_S)} \ge \frac{2n^2+O(n)}{n^2+O(n)} = 2+o(1).$$

◇

In proving Theorem 4 we showed that on a accurately worked out family of trees, `Simple` has a better performance ratio than `Approx-MRN`. In practice, however, the latter is much better than `Simple` (see Section 4). Moreover, it is not difficult to find a different family of trees for which `Simple` is also provably worse.

Given an integer k, let T_k be the tree composed by a root r with k children u_i, each having c leaves except for u_1 having $c+1$ leaves (see Figure 3). The solution $\mathcal{P}_A$ of `Approx-MRN` on such trees is forced to use two sets for the leaves of u_1, but it uses only one set for the other u_is, hence $\rho_{T_k}(u_1, \mathcal{P}_A) = 2$ and $\rho_{T_k}(u_i, \mathcal{P}_A) = 1$, for $1 < i \le k$. At the root we have $\rho_{T_k}(r, \mathcal{P}_A) = k+1$ and for the whole tree $\rho_{T_k}(\mathcal{P}_A) = 2k+2$.

On the other hand, the solution $\mathcal{P}_B$ of `Simple` splits leaves rooted at each u_i into two sets and, thus, $\rho_{T_k}(u_i, \mathcal{P}_B) = 2$ for $1 \le i \le k$ and still $\rho_{T_k}(r, \mathcal{P}_B) = k+1$. For the whole tree we have $\rho_{T_k}(\mathcal{P}_B) = 3k+1$.

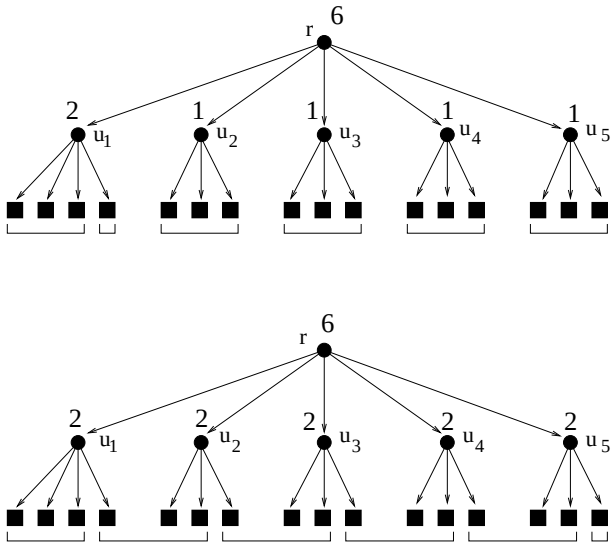

Figure 3: Example of a tree on which `Simple` solution is worst than the `Approx-MRN` one. Numbers close to vertices are the redirections numbers.

4 Experimental results

In order to evaluate the performance of `Approx-MRN` against realistic datasets, we have executed a number of experiments on data taken from various sources on the web. We have also compared its performance with that of `Simple`, which in some cases can be a viable alternative.

4.1 Datasets

We have collected three different datasets: "Open Directory Project" (ODP), "Yahoo!" and "Arianna".

Dataset	*Documents*	*Categories*
ODP	3,266,779	403,550
Yahoo!	2,336,117	102,880
Arianna	54,988	285

Table 1: The datasets.

The first dataset was obtained from a dump of the "Open Directory Project" [13] database (freely available on-line). The ODP powers core directory services for some of the most popular portals and search engines on the web, including AOL Search, Netscape Search, Google, Lycos, DirectHit, and HotBot.

The second dataset was obtained by recursively downloading the first 5 levels of the Yahoo! hierarchy [14] and using the information provided in the pages at the 5-th level to estimate the contents of the remaining levels of the hierarchy.

The last dataset was kindly provided by the staff of "Arianna" [15], an Italian search engine and web directory service.

Table 1 summarizes the number of documents (leaves) and categories (internal nodes) for every dataset.

4.2 Algorithms

We implemented two variants of `Approx-MRN` choosing two different implementations for the BP subroutine (that, as one can easily check, are both reasonable in the sense of Section 3).

The first, which we call "Half Full Fit" (`HFF`), is based on a straightforward on-line algorithm that keeps one open set at a time and packs items until at least half the capacity is used (possibly joining the two sets with smallest level when all items have been taken care of).

The second, which we call "First Fit Decreasing" (`FFD`) [9], is based on the well known First Fit Decreasing Bin Packing strategy.

We also implemented `Simple`, that, as already pointed out, just packs the leaves traversing the frontier of the tree from left to right using for every set all the available capacity.

We use `Simple` mainly as a benchmark to test the effectiveness of the more complicated strategies of Section 3.

4.3 Performance evaluation

We run `HFF`, `FFD` and `Simple` on the three datasets for different capacity c. We choose the values of c according to the number of documents in the datasets, so that the resulting number of sets in the partition (number of needed servers) ranges from tens to thousands.

For every run, we computed the following measures:

1. the number $|\mathcal{P}|$ of sets in the partition,
2. a measure of extra space used, defined as $\kappa = |\mathcal{P}|/\lceil |L|/c \rceil$,
3. the total number of redirections $\rho(\mathcal{P})$,
4. the average number of extra redirections $\bar{\chi}(\mathcal{P})$.

Observe that the first two measures are related to the space efficiency, while the last two to the query efficiency.

Capacity	*Algorithm*	$\lvert\mathcal{P}\rvert$	κ	$\rho(\mathcal{P})$	$\bar{\chi}(\mathcal{P})$
4000	`HFF`	1,132	0.3856	407,884	0.0107
	`FFD`	827	0.0122	406,767	0.0080
	`Simple`	817	0.0000	409,300	0.0142
8000	`HFF`	564	0.3790	405,500	0.0048
	`FFD`	413	0.0098	405,005	0.0036
	`Simple`	409	0.0000	406,416	0.0071
16000	`HFF`	280	0.3659	404,414	0.0021
	`FFD`	207	0.0098	404,202	0.0016
	`Simple`	205	0.0000	404,988	0.0036
32000	`HFF`	144	0.3981	403,943	0.0010
	`FFD`	104	0.0097	403,840	0.0007
	`Simple`	103	0.0000	404,268	0.0018
64000	`HFF`	74	0.4231	403,727	0.0004
	`FFD`	53	0.0192	403,676	0.0003
	`Simple`	52	0.0000	403,906	0.0009
128000	`HFF`	33	0.2692	403,617	0.0002
	`FFD`	27	0.0385	403,601	0.0001
	`Simple`	26	0.0000	403,731	0.0004

Table 2: Experiments on the ODP dataset.

Capacity	*Algorithm*	$\lvert\mathcal{P}\rvert$	κ	$\rho(\mathcal{P})$	$\bar{\chi}(\mathcal{P})$
4000	`HFF`	836	0.4291	106,769	0.0378
	`FFD`	587	0.0034	105,573	0.0262
	`Simple`	585	0.0000	106,828	0.0384
8000	`HFF`	410	0.3993	104,606	0.0168
	`FFD`	294	0.0034	104,095	0.0118
	`Simple`	293	0.0000	104,852	0.0192
16000	`HFF`	200	0.3605	103,623	0.0072
	`FFD`	148	0.0068	103,423	0.0053
	`Simple`	147	0.0000	103,871	0.0096
32000	`HFF`	101	0.3649	103,203	0.0031
	`FFD`	74	0.0000	103,120	0.0023
	`Simple`	74	0.0000	103,376	0.0048
64000	`HFF`	50	0.3514	103,018	0.0013
	`FFD`	37	0.0000	102,985	0.0010
	`Simple`	37	0.0000	103,128	0.0024
128000	`HFF`	24	0.2632	102,934	0.0005
	`FFD`	19	0.0000	102,922	0.0004
	`Simple`	19	0.0000	103,003	0.0012

Table 3: Experiments on the Yahoo! dataset.

Capacity	*Algorithm*	$\lvert\mathcal{P}\rvert$	κ	$\rho(\mathcal{P})$	$\bar{\chi}(\mathcal{P})$
125	`HFF`	758	0.7227	2,602	8.1298
	`FFD`	461	0.0477	1,684	4.9088
	`Simple`	440	0.0000	1,784	5.2596
250	`HFF`	329	0.4955	1,182	3.1474
	`FFD`	239	0.0864	928	2.2561
	`Simple`	220	0.0000	1,032	2.6211
500	`HFF`	153	0.3909	646	1.2667
	`FFD`	117	0.0636	565	0.9825
	`Simple`	110	0.0000	656	1.3018
1000	`HFF`	76	0.3818	441	0.5474
	`FFD`	58	0.0545	404	0.4175
	`Simple`	55	0.0000	469	0.6456
2000	`HFF`	39	0.3929	355	0.2456
	`FFD`	29	0.0357	337	0.1825
	`Simple`	28	0.0000	377	0.3228
4000	`HFF`	19	0.3571	313	0.0982
	`FFD`	15	0.0714	307	0.0772
	`Simple`	14	0.0000	331	0.1614

Table 4: Experiments on the Arianna dataset.

4.4 Discussion

For what concerns the measure κ, `Simple` obviously achieves $\kappa = 0$ while `FFD` is usually slightly better than `HFF`. On the other hand, both `FFD` and `HFF` give better figures for $\bar{\chi}_T(\mathcal{P})$ than `Simple`; also in this case `FFD` is slightly preferable over `HFF`.

Empirical evidence seems to suggest that allowing for very little extra space, can be instrumental to achieving a dramatic improvement for the average number of extra redirections. For example, figures in Table 2 for capacity 128,000, show that an increase of 4% in the size leads to a 400% improvement in the average number of extra redirections, while figures in Table 3 (for the same capacity), show that even with the same number of servers used one can have a 300% improvement.

In a system where queries are in the order of thousands per second, this can lead to great advantages.

4.5 Implementation details

The measure $\rho(\mathcal{P})$ can be computed efficiently according to the following scheme.

Number the leaves of the tree according to a post-order visit. Now visit again the tree and,

- if you are at a leaf, mark with its number the set of the partition containing it,
- if you are at an internal node v, first visit all of its children, and then count how many sets in the partition have a mark between the minimum and maximum leaf number of the subtree $S(v)$. As one can check, such value is $\rho_T(v, \mathcal{P})$.

If the data structure representing a leaf of the tree maintains a pointer to the partition containing it, marking requires constant time for every leaf, while counting the number of sets in $\mathcal{P}$ with a number within a given interval can be done in $O(|\mathcal{P}|)$ time. Hence, the measure $\rho_T(\mathcal{P})$ can be computed in $O(|T||\mathcal{P}|)$ time (and linear space).

For what concerns the implementation of `Approx-MRN`, in order to achieve a reasonable running time, we had to employ a slightly more complicated data structure.

We represented the partition of the leaves by means of a Union-Find data structure endowed with an additional doubly linked list (dll) that keeps track of the root of every tree in the Union-Find forest, i.e., of the sets in the partition. Updating the dll after a union operation requires extra constant time with respect to the usual Union-Find implementation.

We can take advantage of the dll to implement the `Approx-MRN` very efficiently. Indeed, for every recursive call, with small extra effort, one can maintain the head and tail of the sub-dll corresponding to the collection of sets in input to the BP subroutine, instead of scanning every time the Union-Find forest to look for the roots.

5 Conclusions and further work

We believe that a careful investigation of the computational problems underlying the efficient management of a hierarchical document space is worthwhile. In particular it becomes crucial if one aims at fully exploiting hierarchical systems on large scale data sets. This paper is a first step towards this goal, and has provided some guidelines concerning potential advantages of a well designed distribution of the document collection among the servers.

A possible direction for further investigations on the subject of this paper is the study of the impact of different (i.e., non uniform) distributions of user's query on the measure of redirections.

Another obvious direction concerns the addition of incrementality in the document collection under different hypotheses on the ratio of change in the document collection itself.

References

[1] S. Chakrabarti, B. Dom, R. Agrawal, and P. Raghavan, Scalable feature selection, classification and signature generation for organizing large text databases into hierarchical topic taxonomies, *The VLDB Journal* 7 (1998), 163–178.

[2] C. Chekuri, M. H. Goldwasser, P. Raghavan, and E. Upfal, Web Search Using Automatic Classification, *Proc. Sixth Int. World Wide Web Conference*, Santa-Clara, CA, 1997.

[3] P. Crescenzi, V. Kann, A compendium of NP optimization problems, `http://www.nada.kth.se/theory/problemlist.html`.

[4] I. S. Dhillon, J. Fan, and Y. Guan, Efficient clustering of very large document collections, *Data Mining for Scientific and Engineering Applications*, R. Grossman, G. Kamath, and R. Naburu editors, Kluwer Academic Publishers, 2001.

[5] I. S. Dhillon and D. S. Modha, A data clustering algorithm on distributed memory multiprocessing, *Large-Scale Parallel Data Mining, Lecture Notes in Artificial Intelligence*, Volume 1759, pp. 245-260, 2000 (also appears *Proc. Large-scale Parallel KDD Systems Workshop, ACM SIGKDD*, 1999).

[6] S. Dumais and H. Chen, Hierarchical Classification of Web Content, *Proceedings of the 23rd ACM SIGIR*, Athens, Greece, 2000.

[7] O. Frieder, D. Chowdhury, D. A. Grossman, and G. Frieder, Efficiency Considerations for Scalable Information Retrieval Servers, *Journal of Digital information*, 1 (2000). Available at `http://jodi.ecs.soton.ac.uk/Articles/v01/i05/Frieder/`.

[8] M. R. Garey, and D. S. Johnson, Computers and Intractability: a guide to the theory of NP-completeness, W. H. Freeman Company, San Francisco, 1979.

[9] D. S. Johnson, A. Demers, J. D. Ullman, M. R. Garey, and R. L. Graham, Worst-case performance bounds for simple one-dimensional packing algorithms, *SIAM J. of Computing* 3 (1974), 299-325.

[10] N. Karmarkar, R. M. Karp, An efficient approximation scheme for the one-dimensional bin packing problem, *Proc.* 23^{rd} *Ann. IEEE Symp. on Foundations of Computer Science*, 1982, 312-320.

[11] D. Simchi-Levi, New worst case results for the bin packing problem, *Naval Res. Logistics* 41 (1994), 579-585.

[12] R. Weiss, B. Velez, M. Sheldon, C. Namprempre, P. Szilagyi, A. Duda, and D. Gifford, Hypursuit: A hierarchical network search engine that exploits content-link hypertext clustering, *Proceedings of the Seventh ACM Conference on Hypertext*, Washington USA, 1996.

[13] `http://www.dmoz.org/`

[14] `http://www.yahoo.com/`

[15] `http://arianna.libero.it/`