OPTIMIZING WEB SERVER'S DATA TRANSFER WITH HOTLINKS

Evangelos Kranakis
School of Computer Science, Carleton University
Ottawa,ON. K1S 5B6 Canada
kranakis@scs.carleton.ca

Danny Krizanc
Department of Mathematics, Wesleyan University
Middletown, CT. 06459 USA
dkrizanc@wesleyan.edu

Miguel Vargas Martin
School of Computer Science, Carleton University
Ottawa,ON. K1S 5B6 Canada
mvargas@scs.carleton.ca

ABSTRACT

We study the optimization of the expected number of bytes that must be transferred by the Web server when a user visits one of its pages. Given a Web site, we want to find an assignment of hotlinks (shortcuts) that minimizes the expected data transfer, i.e., the average amount of bytes to be downloaded in order to reach one of its pages. We propose an optimization algorithm and perform simulations in real and random Web sites. The simulations reveal that the average data transfer can be reduced by at least 11% and as much as 30%.

KEYWORDS

Web optimization, hotlinks, data transfer, Web modeling, random Web sites.

1. INTRODUCTION

One factor that slows down Web performance is the inevitable downloading of information that does not interest users. This problem happens to anyone who surfs the Web searching for information. For example, suppose we are on page *v* and want to get the information located in page *w*. Among the hyperlinks of *v*, however, there is no hyperlink *(v, w)* that takes us directly to *w*. Inevitably, we have to "download" other pages until we find a page *x* with the desired hyperlink *(x, w)*. Given a Web site, we want to find an assignment of hotlinks (shortcuts) that minimizes the expected data transfer, i.e., the necessary amount of bytes to be downloaded to reach one of its pages.

Previous work concerned the problem of improving Web performance by optimizing the average number of steps required to reach the pages of a Web site. As a consequence of minimizing the number of steps, we certainly would reduce the amount of irrelevant information that needs to be transferred, yet, that would not be the aim. In this paper we study the assignment of hotlinks for optimizing the average data transfer. We restrict the problem to assigning at most one hotlink per node.

2. NOTATION AND TERMINOLOGY

A good way of measuring the access cost of a Web site is to consider the average data transfer generated for reaching a Web page. Under this perspective, we define the *weight* (in bytes) of a page v, w_v, as its own size plus the size of its embedded files. The *access weight of a page* v, $w(v)$, is equal to the sum of the weights of the pages contained in the shortest path between the home page and v. Consider a tree T with root r and m leaves (numbered 1,2,…, m), obtained by performing a breadth first search of a Web site starting from the home page r. Page v is a *descendant* of page u if there is a path of hyperlinks going from u to v in T, and we say that v is at a higher level than u.

Every node u of the tree has a weight w_u associated to it. To reach the Web page at u from the home page we must download all the Web pages in the path from the home page to u. Hence, if $\Pi(r, u)$ is the shortest path from the home page r to u then we have that the *access weight*, w_u, at u is given by the formula

$$w(u) = \sum_{v \in \Pi(r,u)} w_v$$

Without loss of generality, we may normalize the costs w_v.

At the same time, we have the access probabilities p_u at the nodes. This access probability is the frequency with which the node u may be visited from the home page. We have that

$$p_u = \sum_{v \text{ child of } u} p_v$$

The *access cost* of page v is $c(v) = w(v) \cdot p_v$.

The *expected data transfer* for accessing a random leaf page from the root of a tree T with a probability distribution p on its leaves is given by the following equation.

$$E_w[T, p] = \sum_{u \text{ is a leaf}} w(u) \cdot p_u$$

A hotlink $h = (x, y)$, is an extra hyperlink that provides a shortcut from page x to page y, such that y is a descendant of x. We say that the *hyperparent* of y is x.

Given a hotlink assignment $H = \{h_1,..., h_k\}$ over T, the resulting Web site is denoted T^H. Define $\Pi^H(r, u)$ as the path from r to u in T^H. Let $w^H(u)$ denote the access weight of page u in T^H. The gain of H over the expected data transfer of T is denoted $\mathbf{G}(H)$. We have the following:

$$\begin{aligned} w^H(u) &= \sum_{v \in \Pi^H(r,u)} c_v \\ E_w[T^H, p] &= \sum_{i=1}^{m} w^H(i) \cdot p_i \\ \mathbf{G}(H) &= E_w[T, p] - E_w[T^H, p] = \sum_{i=1}^{m} (w(i) - w^H(i)) \cdot p_i \end{aligned} \quad (1)$$

The *weighted hotlink assignment problem* consists in maximizing Equation 1 by assigning at most one hotlink per page. This problem has been proven to be NP-hard for the case of directed acyclic graphs (Vargas Martin, 2002). Vargas Martin also studies a variation of the problem consisting of assigning at most k hotlinks per page.

3. ALGORITHM *weighted-greedyBFS*

In order to solve the weighted hotlink assignment problem, one is tempted to pursue a similar idea to that of Kranakis et al., 2001 in order to find a balanced partitioning of the tree. Inevitably this will also lead to a recursive algorithm like the one presented by Kranakis et al. A complete mathematical analysis of the problem is presented by Vargas Martin, 2002 and Czyzowicz et al., 2003. In this section we present a heuristic algorithm called *weighted-greedyBFS*.

Algorithm *weighted-greedyBFS* described in Algorithm 1, operates in the same way as algorithm *greedyBFS* presented by Czyzowicz et al., 2001. The difference between these two algorithms is that the former attempts to maximize Equation 1 in terms of data transfer, whereas the latter in terms of number of hops. Algorithm *weighted-greedyBFS* assigns hotlinks iteratively in breadth first search order beginning with the home page. Consider a hotlink *(s, t)*. In each iteration of the algorithm, *s* corresponds to the next node in breadth first search order, and *t* corresponds to the descendant of *s* that maximizes the gain, but that is not a descendant of a node *x*, which is at a higher level than *s* and already has an incoming hotlink. Function *next_in_BFS_order* returns each node of the tree in breadth first search order, starting from the home page. The algorithm stops when no more hotlinks can be assigned.

Algorithm 1 *weighted-greedyBFS*(*T*)

1. $H = \phi$ *// H is initially empty, and it will hold the set of hotlinks that is returned in Step 3.*
2. *while(* $(s = next\_in\_BFS\_order) \neq \phi$ *)*
   - 2.1. $t = v$: *v maximizes Equation* 1*; and v is a descendant of s; and v does not have a hyperparent; and v is not a descendant of a node x, which is at a higher level than s and already has a hyperparent.*
   - 2.2. *if* $t \neq \phi$ *then* $H = H \cup \{(s, t)\}$
3. *return H*

4. SIMULATIONS AND CASE STUDY

In this section we use simulations to evaluate algorithm *weighted-greedyBFS*. The algorithm is evaluated on three different kinds of structures: random Web sites, real Web sites with an unknown access probability distribution, and an actual Web site with a known access probability distribution.

Czyzowicz et al., 2001 study the simulation of a random Web site along with the popularity of its pages. In these simulations, we need to assign file sizes (in bytes) to the pages of the random Web sites. Recall that the size of a page is equal to its own size plus the size of its embedded files. We associate a random size to each page and obtain 95% confidence intervals for all the results that we present. Vargas Martin, 2002, validates the correctness of the simulations by showing that the file sizes of the random Web sites actually resemble the theoretical file sizes distribution.

4.1 Web sites used for simulation

For the simulations, we use the same kind of Web sites discussed by Czyzowicz et al., 2001, namely, random Web site trees and real Web sites. To test algorithm *weighted-greedyBFS* we need to enhance those Web sites by associating weights to the Web pages. The weight of a Web page represents the size in bytes of the Web page plus its embedded files, e.g., images, audio, etc. An interesting question arises: What is the distribution of file sizes on the Web?

Arlitt and Williamson, 1997 carefully analyse the traffic of a Web server. They prove experimentally that the file size distribution is *heavy tailed* (see Resnick, 1997 for a background of heavy tail models). The assertion of Arlitt and Williamson was confirmed by Crovella and Bestavros, 1997, who show experimentally that file sizes greater than about 1,000 bytes can be well modeled with a Pareto distribution. Barford and Crovella, 1998 show experimentally that file sizes can be modeled with a hybrid distribution. Files of “small” size can be well modeled with a lognormal distribution, whereas “big” files can be modeled

with Pareto distribution. According to Barford and Crovella, the body of the distribution is modeled with lognormal distribution for values smaller than 133 KBytes. Downey, 2001, creates a model that suggests that file sizes are modeled only by a lognormal distribution. Mitzenmacher, 2002, points out why the arguments of Downey yield only a lognormal distribution. Mitzenmacher corroborates the results of Barford and Crovella and suggests a double Pareto distribution for the body of the curve and a Pareto distribution for the tail.

Assuming that the model proposed by Barford and Crovella, 1998, is correct, we create a file size distribution as follows:

$$P[w_v = x] = \begin{cases} \frac{1}{x\sigma\sqrt{2\pi}} e^{-(\ln x - \mu)^2/2\sigma^2} & \text{if } x < \text{cutoff} \\ \alpha k^\alpha x^{-(\alpha+1)} & \text{otherwise,} \end{cases} \quad (2)$$

where w_v is the size in bytes of page v and its embedded files. The values of μ, σ and α are taken from the observations of Barford et al., 1999 on the requests of over 40,000 files from over 500 users in 1998. The values of k and the *cutoff point* are calculated to fit the results of Barford et al., 1999 who observe that 83% of the files fall in the body of the distribution. The values used in Formula 2 are displayed in Table 1.

Table 1. Parameters used in Formula 2

Parameter	Value
μ	7.796
σ	1.625
k	8,863
α	1.47
cutoff point	10,790

For the case of random Web sites, we assign weights to the pages in a random fashion, according to the distribution of Formula 2.

The straightforward solution for getting the actual weights of real Web sites would be to download the entire Web sites, however this action would require enormous storage capacity. One alternative would be to process the information contained in the log files, however the log files are usually undisclosed to the public. Therefore we use random file sizes. We search the number of embedded files on every page and for each of them we withdraw a size from the distribution given by Formula 2.

4.2 Results of the simulations

We simulate the operation of algorithm *weighted-greedyBFS* on random Web sites. The maximum proportion of gain on the access cost is 30%, whereas the minimum is 11%.

4.2.1 Algorithm *weighted-greedyBFS* evaluated on random Web sites

The results of the simulations are plotted in Figure 1. We plot the average proportion of gain that can be attained by the algorithm. The average proportion of gain has a 95% confidence interval of at most ±2.27. Observe that the gain of the algorithm oscillates between narrow intervals, indicating that the size of the tree does not greatly affect the performance.

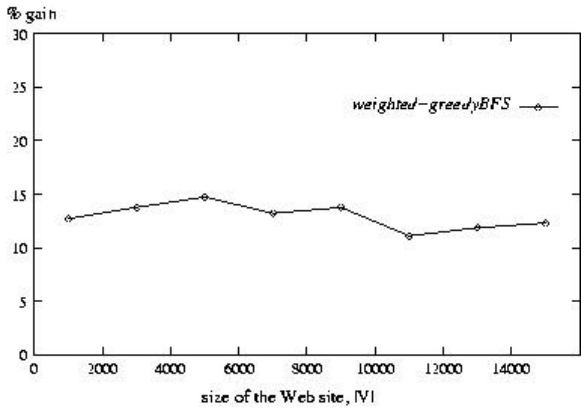

Figure 1. Gain obtained by applying *weighted-greedyBFS* to randomly generated Web sites of size 1,000 to 15,000. The average proportion of gain has a 95% confidence interval of at most ±2.27

4.2.2 Algorithm *weighted-greedyBFS* evaluated on real Web sites

We show the results of the simulations in Figure 2. The average number of nodes in the sample of Web sites is 9,669.2. Observe that the proportion of gain stabilizes after assigning less than 500 hotlinks, which is roughly 10% of the average number of Web pages.

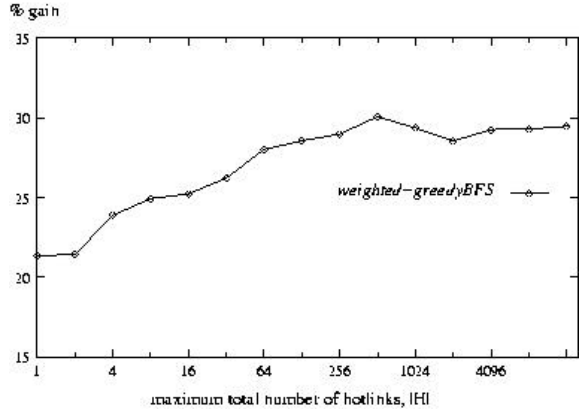

Figure 2. Average gain obtained by applying *weighted-greedyBFS* to actual Web sites. The *x* axis are plotted in logarithmic scale base 2. The average proportion of gain for each of the eleven Web sites has a 95% confidence interval of at most ±7.40

4.2 Case study

We are able to test our algorithms on the *scs.carleton.ca* domain because we have all the necessary information, i.e., the hyperlink structure, access logs, and file sizes.

The case study is conducted in a similar way as by Czyzowicz et al., 2001. In this case, we need to associate real weights to the Web pages. The weight of a Web page is its own size plus the size of its embedded files (in bytes).

Figure 3 illustrates the proportion of gain obtained by the algorithm. The *scs.carleton.ca* domain contains 798 pages. Observe that the maximum gain of the algorithm is achieved at $|H| = 53 << 798$, which represents less than 10% of the number of pages in the site.

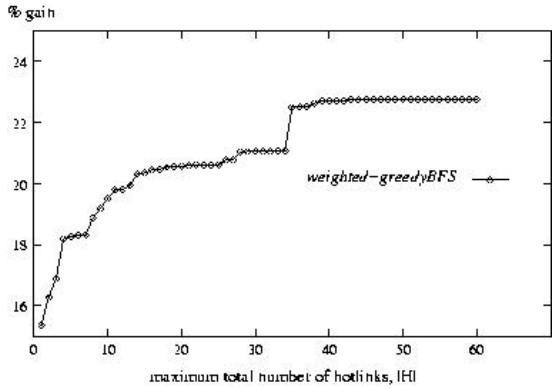

Figure 3. Gain attained by algorithm *weighted-greedyBFS* in the *scs.carleton.ca* domain. The domain contains 798 pages and the maximum gain is attained at $|H| = 52$ hotlinks, which indicates that we can get good gain with just a "few" hotlinks

5. CONCLUSION

We studied the optimization of the expected number of bytes that must be transferred by the Web server when a user visits one of its pages. We found that the data transfer can be reduced by at least 11% and as much as 30%. It would be interesting to see to what extent users actually use the hotlinks in order to determine the real reduction in data transfer once the hotlinks are inserted into the Web pages.

ACKNOWLEDGEMENTS

Evangelos Kranakis, Danny Krizanc and Miguel Vargas Martin were supported in part by MITACS (Mathematics of Information Technology and Complex Systems) NCE (Networks of Centres of Excellence) grant. Evangelos Kranakis was also supported in part by NSERC (Natural Sciences and Engineering Research Council of Canada) grant. Miguel Vargas Martin was also supported in part by CONACYT (Science and Technology Council of Mexico) grant.

REFERENCES

Arlitt, M.F. and Williamson, C.L., 1997. Internet Web Servers: Workload Characterization and Performance Implications. *In IEEE/ACM Transactions on Networking,* Vol. 5, No. 5, pp. 631-645.

Barford, P., Bestavros, A., Bradley, A. and Crovella, M., 1999. Changes in Web Client Access Patterns: Characteristics and Caching Implications. *In World Wide Web,* Vol. 2, No. 1-2, pp. 15-28.

Barford, P. and Crovella, M., 1998. Generating Representative Web Workloads for Network and Server Performance Evaluation. *Proceedings of Measurement and Modeling of Computer Systems.* Madison, USA, pp. 151-160.

Crovella, M.E. and Bestavros, A., 1997. Self-similarity in World Wide Web Traffic. Evidence and Possible Causes. *In IEEE/ACM Transactions on Networking,* Vol. 5, No. 6, pp. 835-846.

Czyzowicz, J., Kranakis, E., Krizanc, D., Pelc, A. and Vargas Martin, M., 2001. Evaluation of Hotlink Assignment Heuristics for Improving Web Access. *In Proceedings of the International Conference on Internet Computing.* Las Vegas, USA, pp. 793-799.

Czyzowicz, J., Kranakis, E., Krizanc, D., Pelc, A. and Vargas Martin, M., 2003. Enhancing Hypelink Structure for Improving Web Performance. *In Journal of Web Engineering*, Vol. 1, No. 2, pp. 93-127.

Downey, A.B., 2001. The Structural Cause of File Size Distributions. *Proceedings of the Ninth International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems.* Cincinnati, USA, pp. 361-370.

Kranakis, E., Krizanc, D. and Shende, S., 2001. Approximate Hotlink Assignment. *Proceedings of the Twelfth Annual International Sysmposium on Algorithms and Computation*, Christchurch, New Zealand, pp. 756-767.

Mitzenmacher, M., 2002. Dynamics Models for File Sizes and Double Pareto Distributions. *Preprint*.

Resnick, S.I., 1997. Heavy Tail Modeling and Teletraffic Data. *In Annals of Statistics*, Vol. 25, No. 5, pp. 1805-1869.

Vargas Martin, M., 2002. *Enhancing Hyperlink Structure for Improving Web Performance*. Ph.D. Thesis, School of Computer Science, Carleton University.