Web Caching With Request Reordering

Tomás Feder * Rajeev Motwani † Rina Panigrahy ‡ An Zhu §

Abstract

Current web caching algorithms process requests in the order of the arrival. While such restriction is inevitable in system paging due to the sequential nature of a program, the HTTP requests are (essentially) independent at a high volume proxy server. This gives a proxy server the flexibility to reorder requests, provided no request is inordinately delayed. The expectation is that reordering requests may lead to better performance. We formulate an online k-reordering problem that captures such phenomenon for unit caches. We give a dynamic programming algorithm to solve the offline case. We give O(1) upper and lower bound on the competitive ratio of the online algorithms. We also generalize this problem to any metric space.

1 Introduction

Current web caching algorithms serve requests in the order of arrival, despite the fact that web requests are essentially independent and better hit rates may be achieved by reordering the requests. (Such phenomenon is even more evident at high volume websites). A natural constraint is that no request should be delayed for long. We propose the online k-reordering problem as follows: given a HTTP request sequence $L_0 = (x_1, x_2, \ldots, x_n)$, the proxy server can reorder the sequence so long as $x_i < x_j$ if $j \geq i + k$. The goal is to minimize the miss rate. We begin the study of such problems by considering a cache of size 1, which is already interesting and non-trivial. Given the ordering L_0 and a distance function $d(i,j)$, the goal is come up with a legal ordering so as to minimize the total distance traveled. In the case of web caching, the distance function is: $d(i,i) = 0, d(i,j) = 1$, or to take into account of web page size $d(i,j) = \frac{1}{2}(w_i + w_j)$, for all $i \neq j$. In general this function could be any metric.

We first consider the offline version of the k-*reordering* problem, where $k \geq 2$. Let r denote the total number of web pages (henceforth, colors). We show this problem can be solved in time $O(nk2^k)$ and $O(nk^{r+1})$. Thus the problem is polynomially solvable when k is at most logarithmic, or when k is arbitrary but r is constant. The k-reordering problem is an instance of constructing a total order while satisfying a set of partial orders. We study the special case where the partial orders are disjoint sets. We show that this problem is NP-complete, even in the special case with $r = 10$. In fact, the problem is as easy or hard to approximate as the shortest common supersequence problem.

We then consider the online version of the k-reordering problem with $d(i,j) = 1$ for $i \neq j$ and give a 2-competitive deterministic greedy online algorithm. We provide a lower bound of 1.5 for deterministic algorithms and 1.25 for randomized algorithms. In order to obtain online algorithms with better performance, we consider the online k-reordering problem with l-lookahead, where $l \geq k$. Here the server is given a larger window, of length l, although only the first k elements of the window can be added to the sequence L, as before. We give a deterministic online algorithm that is $1+O(k/l)$ competitive. We also prove a $1+\Omega(k/l)$ lower bound on the competitiveness of any algorithm. These results apply to the situation where $d(i,j) = \frac{1}{2}(w_i + w_j)$ for $i \neq j$. We give a 3-competitive deterministic algorithm for the k-reordering problem in an arbitrary metric space.

We remark that our problem is similar to the k-client problem [1]. In the k-client problem, requests form k different queues, the server can serve any of the first request of each queue. Alborzi et al gave $2k-1$ competitive online algorithms, while $\frac{\lg k}{2}$ is a lower bound on this ratio [1]. Divakaran and Saks [2] studied an online scheduling problem with job set-ups. Their problem is different from ours in that jobs (not requests) arrives with release time, processing time, and sequence-independent but job-dependent set-up times. The goal is to minimize the maximum flow time. A $O(1)$-competitive online algorithm is presented [2].

2 Offline k-reordering and Ordering

THEOREM 2.1. *The offline k-reordering problem can be solved by dynamic programming in time*

*E-mail: tomas@theory.stanford.edu.

†Department of Computer Science, Stanford University, Stanford, CA 94305. Supported in part by NSF Grant IIS-0118173 and a grant from the Okawa Foundation. E-mail: rajeev@cs.stanford.edu.

‡E-mail: rinap@cisco.com.

§Department of Computer Science, Stanford University, Stanford, CA 94305. Supported by a GRPW fellowship from Bell Labs, Lucent Technologies. E-mail : anzhu@cs.stanford.edu.

$O(nk \sum_{s \le r} \binom{k}{s})$.

Proof Sketch: We can assume that the algorithm always chooses one of the first two colors in the window. This takes time $O(k)$ provided we already know, for each of the r colors, up to which point in the window they have already been included.

The ordering problem for a partial order P that is the disjoint union of linear orders, and with $2r$ colors, encodes the shortest common supersequence problem with an alphabet of size r, by inserting after each color c_i a new color c_i'.

3 Online k-reordering

The greedy online k-reordering algorithm always selects the first color c in the window, and advances the window as much as possible to get all occurrences of c, before switching to a different c' which is now first in the window.

THEOREM 3.1. *The greedy online k-reordering algorithm is 2-competitive, but not α-competitive for any $\alpha < 2$ if k is even.*

Proof Sketch: If the optimal algorithm includes all the vertices of color c in an interval I at the same time, then the greedy algorithm starts at most two intervals for color c at positions inside I, thus at most doubling the cost incurred for each color.

For the lower bound, consider a window of size $k = 2$ and a sequence of colors $(123)^{2l}$. The greedy algorithm will cost $6l$ while the optimum is $3l + 1$.

THEOREM 3.2. *No deterministic online algorithm for the k-reordering problem can be α-competitive for $\alpha <$ 1.5 if k is even; no randomized online algorithm can be α-competitive for $\alpha < 1.25$ if k is even.*

Proof Sketch: Let $k = 2$. We start with either 0100 or 01011, which will take either 2 or 3 steps depending on whether we start with 0 or 1. We continue with the same choice of two sequences for $0', 1'$ then back to $0, 1$, and so on. This proves the $3/2 = 1.5$ ratio. With randomization, we can at best choose to start with 0 or 1 at random with probability $1/2$, giving ratio $5/4 = 1.25$. General case of k even is obtained by replacing 0 and 1 with $k/2$ occurrences of 0s and 1s.

4 Online k-reordering with l-lookahead

THEOREM 4.1. *There is a deterministic online k-reordering algorithm with l-lookahead that achieves a factor of $1 + 16(k/l)$, for $l/8k$ integer.*

Proof Sketch: Partition the sequence into groups of size $l/2$, each of these consisting of t segments of size $4k$, where $t = l/8k$. Within a group, let r_i be the number of distinct colors in the ith segment. A collection of elements of the same color included simultaneously in the sequence must be contained in two consecutive segments. This gives a lower bound of $\sum r_i/2$ for the optimal solution. We can choose to split a group of size $l/2$ by selecting, out of the t segments, the segment with smallest r_j for the dividing point between different dynamic programming solutions. The loss is at most $r_j \le \sum r_i/t = (\sum r_i/2)(16k/l)$.

THEOREM 4.2. *Any deterministic online k-reordering algorithm with l-lookahead has competitive ratio at least $1 + 1/((2l/3k) + 1)$ for even k and $2l/3k$ odd. A randomized algorithm has competitive ratio at least $1 + 1/((4l/3k) + 2)$.*

Proof Sketch: Let $k = 2$ again, and consider the two sequences $(01)^{3t+1}00$ and $(01)^{3t+1}011$. Let $l = 6t + 3$. Whether we start with 0 or with 1 determines cost either $2t + 2$ or $2t + 3$.

5 Online k-reordering for a metric space

We consider the following online algorithm. Start with the minimum length path visiting the elements $x_0, \ldots, x_k$, starting with x_0 and ending with x_k. Then do the same for the elements $x_k, \ldots, x_{2k}$, starting with x_k and ending with x_{2k}, and so on.

THEOREM 5.1. *The above online k-reordering algorithm for a metric space is 3-competitive, but not α-competitive for any $\alpha < 3$ if k is even.*

Proof Sketch: Let L be an optimal k-reordering. Let p_i be the last occurrence in L of an x_j with $j \le ik$. The elements of L after p_{i-1} and up to p_i must contain x_{ik}, and only contain elements x_j with $(i-1)k < j < (i+1)k$. By going from p_{i-1} to p_i, then back to p_{i-1} to get x_{ik}, then back to p_i, we obtain a sequence with the desired elements between $x_{(i-1)k}$ and x_{ik}.

For the lower bound, let $k = 2$. The sequence $(01)^{3t+1}0$ has optimal cost $2t+2$ but incurs $6t+3$ here.

References

[1] H. Alborzi, E. Torng, P. Uthaisombut, and S. Wagner. *The k-client Problem.* 8^{th} Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), 73-82, 1997.

[2] S. Divakaran, and M. Saks. *An Online Scheduling Problem with Job Set-ups.* DIMACS Technical Report, 2000.