Randomized Competitive Algorithms for the List Update Problem

Nick Reingold, Jeffery Westbrook, and Daniel D. Sleator

Abstract. We prove upper and lower bounds on the competitiveness of randomized algorithms for the list update problem of Sleator and Tarjan. We give a simple and elegant randomized algorithm that is more competitive than the best previous randomized algorithm due to Irani. Our algorithm uses randomness only during an initialization phase, and from then on runs completely deterministically. It is the first randomized competitive algorithm with this property to beat the deterministic lower bound. We generalize our approach to a model in which access costs are fixed but update costs are scaled by an arbitrary constant d. We prove lower bounds for deterministic list update algorithms and for randomized algorithms against oblivious and adaptive on-line adversaries. In particular, we show that for this problem adaptive on-line and adaptive off-line adversaries are equally powerful.

Key Words. Sequential search, List-update, On-line algorithms, Competitive analysis, Randomized algorithms.

1. Introduction. Recently much attention has been given to competitive analysis of on-line algorithms [7], [20], [22], [25]. Roughly speaking, an on-line algorithm is c-competitive if, for any request sequence, its cost is no more than c times the cost of the optimum off-line algorithm for that sequence. In their seminal work on competitive analysis [25], Sleator and Tarjan studied heuristics commonly used in system software to maintain a set of items as an unsorted linear list. This problem is called the list update or sequential search problem. The cost of accessing an item is equal to its distance from the front of the list, and the list may be rearranged (at a cost of one per swap of adjacent elements) during the processing of a sequence of requests so that later accesses will be cheaper; for example, a commonly requested item may be moved closer to the front.

Maintaining a dictionary as a linear list is frequently used in practice because of its great simplicity. Furthermore, self-adjusting rules are effective because they take advantage of the locality of reference found in real systems. List update techniques have also been used to develop data compression algorithms [5], as
well as fast and simple algorithms for computing point maxima and convex hulls [3], [14]. For all these reasons, the list update problem has been extensively studied [4], [8], [15], [21], [24]. Sleator and Tarjan [25] demonstrated that the move-to-front algorithm, which uses the simple rule of moving an item to the front of the list each time it is accessed, is 2-competitive. Subsequently Karp and Raghavan (private communication, 1990) noted that no deterministic algorithm for the list update problem can be better than 2-competitive, so in a very strong sense move-to-front is as good as any deterministic on-line algorithm.

A great deal of recent work has focused on the use of randomization to improve—sometimes exponentially—the competitiveness of on-line algorithms [2], [12], [13], [18]. Karp and Raghavan (private communication, 1990) inaugurated the study of randomized list update algorithms by showing a lower bound of 1.18 on the competitiveness of any randomized algorithm. Irani discovered a 1.875-competitive randomized algorithm [16], thus exhibiting the first randomized algorithm to beat the deterministic lower bound.

In this paper we examine the effect of randomization in greater depth. We present a very simple randomized algorithm, BIT, that is 1.75-competitive. Our BIT algorithm is not only simple and fast, but, rather remarkably, it makes random choices only during an initialization phase, using exactly one random bit for each item in the list. From then on BIT runs completely deterministically: to process an access to item x, BIT first complements the bit of x, and then moves x to the front if the bit is 1. We call an algorithm that uses a bounded number of random bits regardless of the number of requests barely random. Such barely random algorithms have practical value since random bits can be an expensive resource. To our knowledge, BIT is the first barely random algorithm for any on-line problem that provably has a better competitive ratio than any deterministic algorithm for that problem. Recently Alon et al. have given a barely random algorithm for k-servers on a circle that is 2k-competitive [1]. However, it is known for k = 2, and conjectured for all k, that a k-competitive deterministic k-server algorithm exists, matching the known lower bound of k [9], [20].

We generalize the BIT algorithm to a family of COUNTER algorithms. Using a COUNTER algorithm, we are able to achieve our best result, a 3/\sqrt{3}-competitive algorithm. We also consider a generalized list update model in which the access cost is the same as the standard model but in which the cost of rearrangement is scaled up by some arbitrarily large value d. (This is a very natural extension, because there is no a priori reason to assume that the execution time of the program that swaps a pair of adjacent elements is the same as that of the program that does one iteration of the search loop.) For arbitrary list length and swap cost d, the best-known deterministic algorithm is 5-competitive. We give a family of COUNTER algorithms that are always better than 2.75-competitive and that in fact become more competitive as d increases. This gives evidence that the scaling conjecture of Manasse et al. [20] may apply to randomized algorithms. This version of the list update problem is similar to the replication/migration problems studied by Black and Sleator [6]. We also show a lower bound of 3 on the competitiveness of any deterministic algorithm in this model, so again our randomized algorithms beat the deterministic lower bound.