Information and Computation ••• (••••) •••–•••

Contents lists available at ScienceDirect

Information and Computation

www.elsevier.com/locate/yinco

On the list update problem with advice

Joan Boyar [a,*], Shahin Kamali [b], Kim S. Larsen [a], Alejandro López-Ortiz [c]

[a] University of Southern Denmark, Department of Mathematics and Computer Science, Campusvej 55, 5230 Odense M, Denmark
[b] Massachusetts Institute of Technology, Computer Science and Artificial Intelligence Laboratory, 32 Vassar Street, Cambridge, MA 02139, USA
[c] University of Waterloo, School of Computer Science, 200 University Avenue West, Waterloo, ON N2L 3G1, Canada

ARTICLE INFO

Article history:
Received 13 June 2013
Available online xxxx

Keywords:
List update
Advice complexity
Competitive analysis
Online algorithms

ABSTRACT

We study the online list update problem under the advice model of computation. Under this model, an online algorithm receives partial information about the unknown parts of the input in the form of some bits of advice generated by a benevolent offline oracle. We show that advice of linear size is required and sufficient for a deterministic algorithm to achieve an optimal solution or even a competitive ratio better than 15/14. On the other hand, we show that surprisingly two bits of advice are sufficient to break the lower bound of 2 on the competitive ratio of deterministic online algorithms and achieve a deterministic algorithm with a competitive ratio of $1.\bar{6}$. In this upper-bound argument, the bits of advice determine the algorithm with smaller cost among three classical online algorithms, Timestamp and two members of the Mtf2 family of algorithms. We also show that Mtf2 algorithms are 2.5-competitive.

1. Introduction

List update is a well-studied problem in the context of online algorithms. The input is a sequence of requests to items of a list; the requests appear in a sequential and online manner, i.e., while serving a request an algorithm cannot look at the incoming requests. A request involves accessing an item in the list.[1] To access an item, an algorithm should linearly probe the list; each probe has a cost of 1, and accessing an item in the ith position results in a cost of i. The goal is to maintain the list in a way to minimize the total cost. An algorithm can make a *free exchange* to move an accessed item somewhere closer to the front of the list. Further, it can make any number of *paid exchanges*, each having a cost of 1, to swap the positions of any two consecutive items in the list.

Similar to other online problems, the standard method for comparing online list update algorithms is competitive analysis. The competitive ratio of an online algorithm A is the maximum ratio between the cost of A for serving any sequence and the cost of Opt for serving the same sequence. Here, Opt is an optimal offline algorithm. It is known that, for a list of length l, no deterministic online algorithm can achieve a competitive ratio better than $2l/(l+1)$ (reported in [22]); this converges to 2 for large lists. There are 2-competitive algorithms (hence best possible online algorithms) for the problem; these include Move-To-Front (Mtf) [30] and Timestamp [2].

* Corresponding author.
E-mail addresses: joan@imada.sdu.dk (J. Boyar), skamali@mit.edu (S. Kamali), kslarsen@imada.sdu.dk (K.S. Larsen), alopez-o@cs.uwaterloo.ca (A. López-Ortiz).

[1] Similar to other works, we consider the *static* list update problem in which there is no insertion or deletion.

http://dx.doi.org/10.1016/j.ic.2016.06.007
0890-5401/

Although competitive analysis has been accepted as the standard tool for comparing online algorithms, there are objections to it. One relevant objection is that assuming a total lack of information about the future is unrealistic in many applications. This is particularly the case for the list update problem when it is used as a method for compression [9]. In this application, each character of a text is treated as an item in the list, and the text as the input sequence which is parsed (revealed) in a sequential manner. A compression algorithm can be devised from a list update algorithm A by writing the access cost of A for serving each character in unary.[2] Hence, the size of the compressed file is roughly equal to the access cost of the list update algorithm. In this application, it is possible to include some partial information about the structure of the sequence (text) in the compressed file, for example, which of three algorithms was used to do the compression. This partial information could potentially be stored using very little space compared to the subsequent savings in the size of the compressed file compared with the original file, due to the availability of the partial information [24].

Advice complexity provides an alternative for the analysis of online problems. Under the advice model, the online algorithm is provided with some bits of advice, generated by a benevolent offline oracle with infinite computational power. This reduces the power of the adversary relative to the online algorithm. Variant models are proposed and studied for the advice complexity model [16,17,13,12]. Here, we use a natural model from [13,12] that assumes advice bits are written once on a tape before the algorithm starts, and the online algorithm can access the tape sequentially from the beginning at any time. The advice complexity of an algorithm is then the worst case number of bits read from the tape, as a function of the length of the input. Since its introduction, many online problems have been studied under the advice model. These include classical online problems such as paging [13,21,25], k-server [17,12,28,20], bin packing [15,7], and various coloring problems [10,18,29].

1.1. Contribution

When studying an online problem under the advice model, the first question to answer is how many bits of advice are required to achieve an optimal solution. We show that advice of size $\textsc{Opt}(\sigma)$ is sufficient to optimally serve a sequence σ, where $\textsc{Opt}(\sigma)$ is the cost of an optimal offline algorithm for serving σ, and it is linear in the length of the sequence, assuming that the length of the list is a constant. We further show that advice of linear size is required to achieve a deterministic algorithm with a competitive ratio better than 15/14.

Another important question is how many bits of advice are required to break the lower bound on the competitive ratio of any deterministic algorithm. We answer this question by introducing a deterministic algorithm that receives two bits of advice and achieves a competitive ratio of at most $1.\bar{6}$. The advice bit for a sequence σ simply indicates the best option between three online algorithms for serving σ. These three algorithms are Timestamp, MTF-Odd (MtfO) and MTF-Even (MtfE). Timestamp inserts an accessed item x in front of the first item y (from the front of the list) that precedes x in the list and was accessed at most once since the last access to x. If there is no such item y or x is accessed for the first time, no items are moved. MtfO (resp. MtfE) moves a requested item x to the front on every odd (resp. even) request to x.

Our results indicate that if we dismiss Timestamp and take the better algorithm between MtfO and MtfE, the competitive ratio of the resulting algorithm is no better than 1.75. We also study the competitiveness of MtfE and MtfO, and more generally any algorithm that belongs to the family of Move-To-Front-Every-Other-Access (also known as Mtf2 algorithms). We show that these algorithms have competitive ratios of 2.5.

2. Optimal solution

In this section, we provide upper and lower bounds on the number of advice bits required to optimally serve a sequence. We start with an upper bound:

Theorem 2.1. *Under the advice model, $\textsc{Opt}(\sigma) - n$ bits of advice are sufficient to achieve an optimal solution for any sequence σ of length n, where $\textsc{Opt}(\sigma)$ is the cost of an optimal algorithm for serving σ.*

Proof. It is known that there is an optimal algorithm that moves items using only a family of paid exchanges called *subset transfer* [26]. In a subset transfer, before serving a request to an item x, a subset S of items preceding x in the list is moved (using paid exchanges) to just after x in the list, so that the relative order of items in S among themselves remains unchanged. Consider an optimal algorithm Opt which only moves items via subset transfer. Before a request to x at index i, an online algorithm can read $i-1$ bits from the advice tape, indicating (bit vector style) the subset which should be moved behind x. Provided with this, the algorithm can always maintain the same list as Opt. The total number of bits read by the algorithm will be at most $\textsc{Opt}(\sigma) - n$. □

The above theorem implies that for lists of constant size, advice of linear size is sufficient to optimally serve a sequence. We show that advice of linear size is also required to achieve any competitive ratio smaller than 15/14.

[2] Encodings other than unary correspond to other cost models for list update, and, naturally, encoding positions in binary would improve the compression [9]. The choice of algorithm is also important and tests indicate that Timestamp may be a better algorithm for this than Mtf [3].

Please cite this article in press as: J. Boyar et al., On the list update problem with advice, Inf. Comput. (2016), http://dx.doi.org/10.1016/j.ic.2016.06.007