Theoretical Computer Science ••• (••••) •••–•••

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Know when to persist: Deriving value from a stream buffer ☆

Konstantinos Georgiou [a,*,1], George Karakostas [b,1], Evangelos Kranakis [c,1], Danny Krizanc [d]

[a] Ryerson University, Dept. of Mathematics, Toronto, Ontario, Canada
[b] McMaster University, Dept. of Computing & Software, Hamilton, Ontario, Canada
[c] Carleton University, School of Computer Science, Ottawa, Ontario, K1S 5B6, Canada
[d] Wesleyan University, Dept. of Math. & Computer Science, Middletown CT, USA

ARTICLE INFO

Article history:
Received 10 October 2016
Received in revised form 22 March 2017
Accepted 9 May 2017
Available online xxxx

Keywords:
Buffer
Competitive ratio
Data stream
Online

ABSTRACT

We consider PERSISTENCE, a new online problem concerning optimizing weighted observations in a stream of data when the observer has limited buffer capacity. A stream of weighted items arrive one at a time at the entrance of a buffer with two holding locations. A processor (or observer) can process (observe) an item at the buffer location it chooses, deriving this way the weight of the observed item as profit. The main constraint is that the processor can only move *synchronously* with the item stream. PERSISTENCE is the online problem of scheduling the processor movements through the buffer so that its total derived value is maximized under this constraint. We study the performance of the straight-forward heuristic *Threshold*, i.e., forcing the processor to "follow" an item through the whole buffer only if its value is above a threshold. We analyze both the optimal offline and Threshold algorithms in the cases where the input stream is either a random permutation, or its items are iid valued. We show that in both cases the competitive ratio achieved by the Threshold algorithm is at least 2/3 when the only statistical knowledge of the items is the median of all possible values. We generalize our results by showing that Threshold, equipped with some minimal statistical advice about the input, achieves competitive ratios in the whole spectrum between 2/3 and 1, following the variation of a newly defined density-like measure of the input. This result is a significant improvement over the case of arbitrary input streams, where we show that no online algorithm can achieve a competitive ratio better than 1/2.

1. Introduction

Suppose that the Automated Quality Control (AQC) of an assembly line has the ability to check all new parts as they enter the assembly line. Every such check increases our quality confidence by a certain percentage, which depends on the nature of the part itself. Now, suppose that the AQC is given the option of a second look at the same part in the next time slot, with a similar increase in our quality confidence. The downsize of this option, is that when the AQC returns to the beginning

☆ This is an extended version of results which originally appeared in AAIM 2016 [12].

* Corresponding author.
E-mail addresses: konstantinos@ryerson.ca (K. Georgiou), karakos@mcmaster.ca (G. Karakostas), kranakis@scs.carleton.ca (E. Kranakis), dkrizanc@wesleyan.edu (D. Krizanc).

[1] Research supported in part by NSERC Discovery grant.

http://dx.doi.org/10.1016/j.tcs.2017.05.021
0304-3975/

of the assembly line, it will have completely missed the part immediately following the one that was double-checked. We are looking for an algorithm to decide whether to take the option or not with every new item. Obviously, a good strategy would strive to look twice at "low-quality" items, since that would imply the greatest increases to our confidence, while "missing" only pristine-looking ones.

This problem falls within the data stream setting: a sequence of input data is arriving at a very high rate, but the processing unit has limited memory to store and process the input. Data stream algorithms have been explored extensively in the computer science literature. Typical algorithms in this area work with only a few passes (often just one) over the data input and use memory space less than linear in the input size. Applications can be found in processing cell phone calls or Internet router data, executing Web searches, etc. (cf. [20,21]).

In this work we study a new online problem in data stream processing with limited buffer capacity. An online stream of items (the parts in our AQC example) arrives (one item at a time) at a buffer with two locations L_0, L_1 (assembly points 1 and 2 respectively in the example above), staying at each location for one unit of time, in this order. A processor/observer (the AQC) can move between the two locations *synchronously*, i.e., its movements happen at the same time as the items move. This means that if the processor is processing (observing) the i-th item in time t at L_0, moving to L_1 will result in processing again the i-th item at L_1 in time $t+1$. On the contrary, if the processor is processing the i-th item in time t at L_1, moving to L_0 will result in processing the $i+2$-th item at L_0 in time $t+1$; the $i+1$-th item has already moved to L_1 and will leave the buffer without the processor ever encountering it! (just like the AQC totally missed a part). We emphasize that we restrict the processor to not even know what item it missed (i.e., cannot "see" into a location other than its current one). Processing the i-th item (either in L_0 or L_1) produces an added value or payoff. The processor has very limited (constant in our results) memory capacity, and cannot keep more than one value (of size no larger than the largest observable value), as well as it can only do comparisons. The problem we address is whether such a primitive processor can have a strategy to persist and observe (if possible) mostly "good values", especially when compared to an optimal algorithm that is aware of the input stream. We call this online problem Persistence, which to the best of our knowledge is also new.

1.1. Related work

There is extensive literature on data stream algorithms. Here the emphasis is on input data arriving at a very high rate and limited memory to store and process the input (thus stressing a tradeoff between communication and computing infrastructure). A general introduction to data stream algorithms and models can be found in [20,21]. Lower bound models for space complexity are elaborated in [3]. In the section on new directions for streaming models, [21] discusses several alternatives for data streams for permutation streaming of non-repeating items [1], windowed streaming whereby the most recent past is more important than the distant past [15], as well as reset model, distributed continuous computation, and synchronized streaming. Applications of data stream algorithms are explored extensively in the computer science literature and can be found in sampling (finding quantiles [14], frequent items [19], inverse distribution [7], and range-sums of items [2]).

Related to our study is the well-known *secretary problem* which appeared in the late 1950s and early 1960s (see [9] for a historical overview of its origins and [11] which discusses several extensions). It is concerned with the optimal strategy or stopping rule so as to maximize the probability of selecting the best job applicant assuming that the selection decision can be deferred to the end. Typically we are concerned with maximizing the probability of selecting the best job applicant; this can be solved by a maximum selection algorithm which tracks the running maximum.

The problem has fostered the curiosity of numerous researchers and studied extensively in probability theory and decision theory. Several variants have appeared in the scientific literature, including on rank-based selection and cardinal payoffs [6], the infinite secretary problem in [13], secretary problem with uncertain employment in [22], the submodular secretary problem in [5], and the temporary secretary problem [10,16], just to mention a few.

The "secretary problem" paradigm has important applications in computer science of which it is worth mentioning the recent work of [4] which studies the relation of matroids, secretary problems, and online mechanisms, as well as [17] which is investigating applications of a multiple-choice secretary algorithm to online auctions. Obviously the secretary problem differs from Persistence in terms of the objective function: in our case the payoff is the sum of processing payoffs, as opposed to the maximum for the secretary problem. The two problems also differ in the synchronicity and location of arrivals, i.e., what can be accessed and how it is accessed. Nevertheless, the two problems share the inherent difficulty of having to make decisions *on the spot* while missing parts of the input altogether. Notably, and as in solutions for various secretary problems, our algorithms rely on threshold-decision choices. Similar algorithms have been observed by cognitive scientists to be implemented in practice in human behavior [18].

1.2. High level summary of our results & outline of the paper

Our primary focus is the study of the Persistence problem, which we formally define in Section 2.1. Our goal is to compare the performance of any primitive (online) algorithm, which is not aware of the input stream, against the optimal offline algorithm. In Section 2.2 we present all such possible primitive algorithms that we call *Threshold*. Subsequently, in Section 2.3 we analyze the performance of any Threshold online algorithm for deterministic input streams. Our findings indicate that simplistic primitive algorithms are actually optimal (among all online solutions), and are off no more than 1/2

Please cite this article in press as: K. Georgiou et al., Know when to persist: Deriving value from a stream buffer, Theoret. Comput. Sci. (2017), http://dx.doi.org/10.1016/j.tcs.2017.05.021