Pattern Matching in Multiple Streams*

Raphaël Clifford[1], Markus Jalsenius[1], Ely Porat[2], and Benjamin Sach[3]

[1] University of Bristol, Department of Computer Science, Bristol, UK
[2] Bar-Ilan University, Department of Computer Science, Ramat-Gan, Israel
[3] University of Warwick, Department of Computer Science, Coventry, UK

Abstract. We investigate the problem of deterministic pattern matching in multiple streams. In this model, one symbol arrives at a time and is associated with one of s streaming texts. The task at each time step is to report if there is a new match between a fixed pattern of length m and a newly updated stream. As is usual in the streaming context, the goal is to use as little space as possible while still reporting matches quickly. We give almost matching upper and lower space bounds for three distinct pattern matching problems. For exact matching we show that the problem can be solved in constant time per arriving symbol and $O(m + s)$ words of space. For the k-mismatch and k-difference problems we give $O(k)$ time solutions that require $O(m + ks)$ words of space. In all three cases we also give space lower bounds which show our methods are optimal up to a single logarithmic factor. Finally we set out a number of open problems related to this new model for pattern matching.

1 Introduction

We introduce a new set of problems centered on pattern matching in multiple streaming texts. In this model, one symbol arrives at a time and is added to the tail of exactly one of s streaming texts. The task at each time step is to report if there is a new match between a fixed pattern P of length m and a newly updated stream. Our interest is in deterministic algorithms with guaranteed worst case time complexity. The goal is to use as little space as possible while still reporting matches quickly.

The problem of pattern matching in a single stream using limited space had a major breakthrough in 2009 when it was shown that exact matching can be performed using only $O(\log m)$ words of space and $O(\log m)$ time [20]. This result was subsequently simplified [10] and then improved [4] to run in constant time per new symbol. To achieve this small space, these methods are however all necessarily randomised and allow some small probability of error. Where neither randomisation nor error is permitted, a straightforward argument shows us that there is no hope of using space sublinear in the pattern size. This follows directly from the observation that we could use such a matching algorithm to reproduce the pattern in its entirety and that it therefore must use at least linear space.

* This work was partially supported by EPSRC.

Where there are multiple streams however the situation is not as clear cut even where no randomisation is allowed. A naive approach would simply be to store s copies of the working space, one for each stream. This will typically then require $\Theta(ms)$ space overall. Where the number of streams s is large, the space usage of such an approach is therefore likely to be prohibitive.

Our contribution is first to show that for three particularly common pattern matching problems, pattern matching in multiple streams can be solved in considerably less than $\Theta(ms)$ space without the use of randomisation. In Section 3 we give a constant time algorithm for exact matching that requires only $O(m+s)$ words of space. In Section 4 we review a recently introduced data structure which allows us to perform longest common extension (LCE) queries between a streaming text and a fixed pattern in constant time and $O(m)$ space. This data structure is then used in Sections 5 and 6 where we give $O(k)$ time and $O(m + ks)$ space solutions to the k-mismatch and k-difference problems. In Section 7 we give almost matching space lower bounds for our problems as well as lower bounds for some other common pattern matching problems. Finally in Section 8 we set out a number of open problems that immediately arise for this new model of pattern matching.

2 Related Work

Randomised space lower bounds for a wide range of pattern matching problems in a single stream were given in [7]. In [6,9] it was also shown that a large set of pattern matching algorithms could be converted from offline to online with only at worst a multiplicative logarithmic factor overhead in their time complexity. This therefore provided an effective deamortisation of almost the entire field of combinatorial pattern matching and a ready tool for the construction of fast streaming pattern matching algorithms.

In the more usual offline setting, a great deal of progress has been made in finding fast algorithms for a variety of approximate matching problems. One of the most studied is the Hamming distance which measures the number of single character mismatches between two strings. Given a text of length n and a pattern of length m, the task is to report the Hamming distance at every possible alignment. Solutions running in $O(n\sqrt{m \log m})$ time which are based on repeated applications of the FFT were given independently by both Abrahamson and Kosaraju in 1987 [1,15]. Particular interest has been paid to the bounded variant we also consider called the k-mismatch problem. Here a bound k is given and we need only report the Hamming distance if it is less than or equal to k. If the number of mismatches is greater than the bound, the algorithm need only report that fact and not give the actual Hamming distance. In 1986 Landau and Vishkin gave a beautiful $O(nk)$ time algorithm that is not FFT based and uses constant time lowest common ancestor (LCA) operations on the suffix tree of the pattern and text [18]. This was subsequently improved to $O(n\sqrt{k \log k})$ time by a method based on filtering and FFTs again [3]. A separate line of research considered the question of how to find approximations within a