Information Processing Letters 110 (2010) 1021–1025

Contents lists available at ScienceDirect

Information Processing Letters

www.elsevier.com/locate/ipl

A filtering algorithm for k-mismatch with don't cares

Raphaël Clifford [a,*], Ely Porat [b]

[a] University of Bristol, Dept. of Computer Science, Bristol, BS8 1UB, UK
[b] Bar-Ilan University, Dept. of Computer Science, 52900 Ramat-Gan, Israel

ARTICLE INFO

Article history:
Received 21 September 2009
Received in revised form 3 August 2010
Accepted 24 August 2010
Available online 27 August 2010
Communicated by P.M.B. Vitányi

Keywords:
Algorithms
Design of algorithms
Combinatorial problems
String algorithms
Don't cares
Pattern matching
Filtering
Fast Fourier transforms

ABSTRACT

We present a filtering based algorithm for the k-mismatch pattern matching problem with don't cares. Given a text t of length n and a pattern p of length m with don't care symbols in either p or t (but not both), and a bound k, our algorithm finds all the places that the pattern matches the text with at most k mismatches. The algorithm is deterministic and runs in $\Theta(nm^{1/3}k^{1/3}\log^{2/3} m)$ time.

1. Introduction

We consider approximate string matching under the widely used Hamming distance. In particular our interest is in a bounded version of this problem which we call *k-mismatch with don't cares*. Given a text t of length n and a pattern p of length m, we allow either the pattern or the text (but not both) to contain promiscuously matching don't care symbols. Given a bound k, our algorithm finds all the locations where the pattern matches the text with at most k mismatches. If the distance is greater than k, the algorithm need only report that fact and not give the actual Hamming distance.

The problem of exact string matching is a classic one in computer science whose linear time solutions were first presented in the 1970s [4,15]. Determining the time complexity of exact matching with optional single character *don't care* symbols has also been well studied. Fischer and Paterson [12] presented the first solution based on fast Fourier transforms (FFT) with an $\Theta(n\log m\log|\Sigma|)$ time algorithm in 1974,[1] where Σ is the alphabet that the symbols are chosen from. Subsequently, the major challenge has been to remove this dependency on the alphabet size. Indyk [13] gave a randomised $\Theta(n\log n)$ time algorithm which was followed by a simpler and slightly faster $\Theta(n\log m)$ time randomised solution by Kalai [14]. In 2002, the first deterministic $\Theta(n\log m)$ time solution was given [9] which was then further simplified in [7].

The key observation given by [7] but implicit in previous work is that for numeric strings, if there are no don't care symbols, then for all locations $1 \leqslant i \leqslant n-m+1$ we can calculate

$$\sum_{j=1}^{m}(p_j - t_{i+j-1})^2 = \sum_{j=1}^{m}\left(p_j^2 - 2p_j t_{i+j-1} + t_{i+j-1}^2\right) \qquad (1)$$

* Corresponding author.
E-mail addresses: clifford@cs.bris.ac.uk (R. Clifford), porately@cs.biu.ac.il (E. Porat).

0020-0190/$ – see front matter
doi:10.1016/j.ipl.2010.08.012

[1] Throughout this paper we assume the word RAM model with multiplication when giving the time complexity of the FFT. This is in order to be consistent with the large body of previous work on pattern matching with FFTs.

in a total of $\Theta(n \log m)$ time using FFTs. Here p_j indicates the jth symbol in the input string p. The notation holds similarly for t_i in string t. Wherever there is an exact match this sum will be exactly 0. If p and t are not numeric, then an arbitrary one-to-one mapping can be chosen from the alphabet to the set of positive integers $\mathbb{N}$. In the case of matching with don't cares, each don't care symbol in p or t is replaced by a 0 and the sum is modified to be

$$\sum_{j=1}^{m} p'_j t'_{i+j-1} (p_j - t_{i+j-1})^2$$

where $p'_j = 0$ ($t'_i = 0$) if p_j (t_i) is a don't care symbol and 1 otherwise. This sum equals 0 if and only if there is an exact match with don't cares and can also be computed in $\Theta(n \log m)$ time using FFTs.

Approximate matching is one of the fundamental tools of large scale data processing and is both widely studied and used in practice. Errors or noise can occur in data in a great variety of forms and so any search tool on real data must be able to handle a level of approximation. In other cases the data themselves are accurate but we might be tasked for example, with searching for similarity between images in a library or in the case of bioinformatics, a key operation is to search for functional similarities between genes or proteins. In some cases elements of the data are simply unknown and should not be counted towards any measure of similarity. Such data are often represented by don't care or wildcard symbols. As an example in image processing, a rectangular image segment may contain a facial image and the objective is to identify the face in a larger scene. However, background pixels around the faces may be considered to be irrelevant for facial recognition and these should not affect the search algorithm. Alternatively, a consensus sequence derived from multiple alignment in computational biology (see e.g. [11]) may contain unknown values and the aim is to perform approximate matching rapidly on a large DNA or protein database without penalising any matches to the unknown values. Due to the asymmetry between query and database or pattern and text it is often the case that uncertainty lies in either the pattern or the text but not both. It is this model of approximate matching with don't cares in either the pattern or text that we consider here.

In Section 2 related and previous work is discussed. We then formalise the pattern matching problem description and give basic definitions that will be used throughout in Section 3. In Section 4 we explain in detail why don't cares cause problems for traditional filtering algorithms and present our main solution. Finally, in Section 5 we conclude and discuss the open problems that remain to be solved.

2. Related and previous work

Much progress has been made in finding fast algorithms for the k-mismatch problem *without* don't cares over the last 20 years. $\Theta(n\sqrt{m \log m})$ time solutions to the k-mismatch problem based on repeated applications of the FFT were given independently by both Abrahamson and Kosaraju in 1987 [1,16]. Their algorithms are in fact independent of the bound k and report the Hamming distance at every position irrespective of its value. In 1985 Landau and Vishkin gave a beautiful $\Theta(nk)$ algorithm that is not FFT based which uses constant time LCA operations on the suffix tree of p and t [18]. This was subsequently improved to $\Theta(n\sqrt{k \log k})$ time by a method based on filtering and FFTs again [3]. Approximations within a multiplicative factor of $(1+\epsilon)$ to the Hamming distance can also be found in $\Theta(n/\epsilon^2 \log m)$ time [13]. A variant of the edit-distance problem (see e.g. [17]) called the k-difference problem with don't cares was considered in [2]. Progress has also been made recently on the related problem of indexing with errors and don't cares [8,5].

To the authors' knowledge, no non-naive algorithms have been given to date for the k-mismatch pattern matching problem with don't cares. However, the $\Theta(n\sqrt{m \log m})$ divide and conquer algorithms of Kosaraju and Abrahamson [1,16] can be easily extended to handle don't cares in both the pattern and text without changing the overall time complexity. This is because the algorithm counts matches and not mismatches. First we count the number of non-don't care matches in $\Theta(n\sqrt{m \log m})$ time. Then we need only subtract this number from the maximum possible number of non-don't care matches at a particular position in the text in order to count the mismatches. When don't cares are only allowed in only one of the pattern or text this can be computed in linear time as we will show in Section 4.

3. Problem definition and preliminaries

Let Σ be a set of characters which we term the *alphabet*, and let ϕ be the don't care symbol. Let $t = t_1 t_2 \ldots t_n \in \Sigma^n$ be the text and $p = p_1 p_2 \ldots p_m \in \Sigma^m$ the pattern. Either the pattern or the text may also include ϕ in their alphabet but not both. The terms *symbol* and *character* are used interchangeably throughout. Similarly, we will sometimes refer to a *location* in a string and synonymously at other times the *position*. The term *alignment* will only be used to refer to a position in the text.

- Define $HD(i)$ to be the number of mismatches or Hamming distance between p and $t[i, \ldots, i+m-1]$ and define the don't care symbol to match any symbol in the alphabet.
- Define

$$HD_k(i) = \begin{cases} HD(i) & \text{if } HD(i) \leqslant k, \\ \perp & \text{otherwise.} \end{cases}$$

- We say that at position i in t, p is a k-mismatch if $HD_k(i) \neq \perp$.
- Symbols other than don't cares are said to be *solid*. We say that a match between two solid symbols is a *solid match* and we say that a position in the pattern or text corresponding to a solid symbol is a *solid position*.

Our algorithms make extensive use of the fast Fourier transform (FFT). An important property of the FFT is that in the RAM model, the cross-correlation,