Approximate Subset Matching with Don't Cares

Amihood Amir* Moshe Lewenstein* Ely Porat*

1 Introduction

The need for the general task of "pattern matching" is ubiquitous. However, the combinatorial model of the problem varies with the different application domain. This variability is manifested in a number of research directions:

The Object Space: Text Editing motivates that the text and pattern be *strings*. Image processing motivates that the text and pattern be two dimensional arrays. In *tree pattern matching* [8], that was used for solving problems with grammars, the text and pattern are trees. Cole and Hariharan introduced the *Subset Matching problem* where the text and patterns are arrays of sets [5]. The subset matching problem is an interesting generalization of string matching and has proven useful in producing more efficient solutions for tree matching.

The Matching Relation: Various applications require a different notion of "match". This led to the concept of *generalized matching*. Examples are exact matching, parameterized matching [4], matching with "don't cares", less-than matching, and swapped matching.

Approximation: In approximate matching, one defines a distance function between the objects and seeks all text location where the pattern matches the text by a pre-specified "small" distance.
One of the earliest and most natural metrics is the *Hamming distance*, where the distance between two strings is the number of mismatching characters.

The goal of this paper is to investigate the complexity of approximate matching with don't cares in *Subset Matching*. The input of this problem is a text array of n sets totaling s elements and a pattern array of m sets totaling s' elements. There is a match of the pattern in a text location if every pattern set is a subset of the corresponding text set. An efficient algorithm for the approximate subset matching problem with don't cares will both advance our understanding of the subset matching paradigm and could provide tools for more efficient algorithms for other problems, e.g. swap matching or 2-dim parameterized matching. Both of these problems can be solved using approximate subset matching.

Consider the case of subset matching with don't cares. Current methods are not useful in handling "don't care"s in the text. Conceptually, every "don't care" in the text can be replaced by the universal set, but that may prohibitively increase the algorithm's complexity. A new method is needed for handling "don't care"s in text sets. An additional new result of this paper, is counting the number of mismatches wherever the subset relation does not hold. This is, in effect, a generalization of Hamming distance to subset matching.

This paper presents a deterministic algorithm that solves the problem in time $O((s+n)\sqrt{s'\log m})$. Note that for the special case of both text and pattern being composed of singletons (the *string matching with mismatches and don't cares* problem) our time complexity matches the currently best known algorithm [1].

2 Definitions and Preliminaries

Cole and Hariharan [5] introduced subset matching and proposed a near-linear time randomized algorithm for the problem. This was subsequently improved in [6] to a deterministic algorithm. We introduce the approximate version with "don't care" symbols.

Let Σ be our alphabet and let $\star \notin \Sigma$. $\star$ will denote our "don't care" symbol. Let $S_1, S_2 \subseteq \Sigma$ or $S_1 = \star$ or $S_2 = \star$. Denote

$$mis(S_1,S_2) = \begin{cases} |S_1 - (S_1 \cap S_2)|, & \text{if } S_1, S_2 \neq \star; \\ 0, & \text{otherwise.} \end{cases}$$

Definition: The *Approximate Subset Matching Problem with don't cares* is defined as follows.
INPUT: Text $T = T_1, T_2, ..., T_n$ of sets $T_i \subseteq \Sigma$ or $T_i = \star$, $i = 1, ..., n$ and pattern $P = P_1, P_2, ..., P_m$ of sets $P_i \subseteq \Sigma$ or $P_i = \star$, $i = 1, ..., m$.
OUTPUT: For every location i, $1 \le i \le n-m+1$, compute $\sum_{\ell=1}^{m} mis(P_\ell, T_{i+\ell-1})$.

The special case where every set in T or P has either a single element or is $\star$ is known as the *string matching with mismatches and "don't care"s* problem. Abrahamson [1] developed a divide-and-conquer algorithm that solves that problem in time $O(n\sqrt{m \log m})$. Variations on the Abrahamson approach were used in [2, 3]. Here we develop a new variation of this technique.

*Bar-Ilan University, Israel; {amir,moshe,porately}@cs.biu.ac.il

3 The Algorithm

We will assume, as in [5, 6], that w.l.o.g. $n = 2m$ and $2m \leq s + s' \leq 4m$. This is true since the text may be cut into n/m overlapping segments, each having length $2m$. For each segment, the sets may be broken down into smaller sets such that the sum of their elements will range between $2m$ and $4m$ elements.

Intuition: Let S_1 and S_2 be sets. We need to compute $mis(S_1, S_2)$. Consider all pairs of elements $\langle a, b\rangle$, $a \in S_1$, $b \in S_2$. Charge such a pair with 1 if $a \neq b$ and with 0 if $a = b$. Let $pairs(S_1, S_2)$ be the sum of the costs of all pairs.

LEMMA 3.1. *Let S_1, S_2 be sets. Then $mis(S_1, S_2) = pairs(S_1, S_2) - (|S_1|(|S_2| - 1))$.*

It is easy to compute, for every i, the sum $\sum_{\ell=1}^{m}(|P_\ell|(|T_{i+\ell-1}| - 1))$ in time $O(n \log m)$ using polynomial convolutions [7]. The challenge is an efficient computation, for every i, of $\sum_{\ell=1}^{m} pairs(P_\ell, T_{i+\ell-1})$.

The problem lies in the fact that these sets may be of size m each, and thus the computation will require $O(m^2)$ per location. Clearly the sum of the pattern sets may now exceed $O(m)$ (but can not be greater than s'). Denote the sum of the sizes of the pattern sets by s''.

Our algorithm has two stages. In the *summing* stage, we count mismatches of block elements using convolutions [7]. In the *correction* stage we count mismatches within blocks. The "don't care"s are handled separately. We outline the details of the algorithm here.

Dividing into Blocks:

We call a symbol that appears in the text at least $f = \frac{m\sqrt{\log m}}{\sqrt{s''}}$ times a *frequent symbol*. We now divide the text alphabet symbols into blocks in the method of Abrahamson [1]. By the construction, we are guaranteed to have no more than $\frac{4m}{f} = \frac{4\sqrt{s''}}{\sqrt{\log m}}$ blocks, each with at most $O(f)$ different elements in each. Note that each character appears in one block only.

Summing Mismatches of Elements in Different Blocks:

We may sum all pairwise differences of elements in *different* blocks by the following $\frac{4\sqrt{s''}}{\sqrt{\log m}}$ convolutions.

1. For each block B do
   a. Create string T' where the ith location of T' is the number of elements in T_i appearing in B.
   b. Create string P' where the ith location of P' is the number of elements in P_i *not* appearing in B.
   c. Compute $R_B \leftarrow T' \times P'^R$.
2. Compute $R \leftarrow \sum_B R_B$.

We now have the pairwise differences of elements in different blocks. We have not counted the differences between pairs of elements in the same block.

Correction within Blocks:

For every one of the s'' elements in the pattern, compare it with all text elements in its block and use the index [the second element in the pair] to add a mismatch to the appropriate mismatch counters.

Time:
$O(m \log m)$ for block construction.
$O(\frac{4\sqrt{s''}}{\sqrt{\log m}} m \log m) = O(m\sqrt{s'' \log m})$ for summing mismatches of pairs in different blocks.
$O(s''f) = O(s'' \frac{m\sqrt{\log m}}{\sqrt{s''}}) = O(m\sqrt{s'' \log m})$ for the correction stage.

Total Time: $O(m\sqrt{s'' \log m})$.

Total Time for General Text and Pattern: $O((s+n)\sqrt{s' \log m})$.

Note that we still need a polynomial multiplication for computing $\sum_{\ell=1}^{m}(|P_\ell|(|T_{i+\ell-1}| - 1))$ to achieve the subset mismatch value from the pairwise mismatch.

3.1 The Deterministic "don't care" case

In the pairwise mismatch computation always put a 0 in the location of a "don't care". In the correction phase always put a 0 in the location of a "don't care".

References

[1] K. Abrahamson. Generalized string matching. *SIAM J. Comp.*, 16(6):1039–1051, 1987.

[2] A. Amir, Y. Aumann, G. Landau, M. Lewenstein, and N. Lewenstein. Pattern matching with swaps. *Proc. 38th IEEE FOCS*, pages 144–153, 1997.

[3] A. Amir, M. Lewenstein, and E. Porat. Faster algorithms for string matching with k mismatches. In *Proc. 11th ACM-SIAM Symp. on Discrete Algorithms (SODA)*, pages 794–803, 2000.

[4] B. S. Baker. A theory of parameterized pattern matching: algorithms and applications. In *Proc. 25th Annual ACM Symposium on the Theory of Computation*, pages 71–80, 1993.

[5] R. Cole and R. Harihan. Tree pattern matching and subset matching in randomized $o(n \log^3 m)$ time. *Proc. 29th ACM STOC*, pages 66–75, 1997.

[6] R. Cole, R. Harihan, and P. Indyk. Tree pattern matching and subset matching in deterministic $o(n \log^3 n)$ time. *Proc. 10th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 245–254, 1999.

[7] M.J. Fischer and M.S. Paterson. String matching and other products. *Complexity of Computation, R.M. Karp (editor), SIAM-AMS Proceedings*, 7:113–125, 1974.

[8] S. R. Kosaraju. Efficient tree pattern matching. *Proc. 30th IEEE FOCS*, pages 178–183, 1989.