Approximate Swapped Matching

Amihood Amir[1], Moshe Lewenstein[2], and Ely Porat[3]

[1] Department of Mathematics and Computer Science,
Bar-Ilan University, 52900 Ramat-Gan, Israel
and Georgia Tech
Tel. (972-3)531-8770;
amir@cs.biu.ac.il

[2] Department of Mathematics and Computer Science,
Bar-Ilan University, 52900 Ramat-Gan, Israel
Tel. (972-3)531-8407;
moshe@cs.biu.ac.il

[3] Department of Mathematics and Computer Science,
Bar-Ilan University, 52900 Ramat-Gan, Israel
and Weizmann Institute.
Tel. (972-3)531-8407;
porately@cs.biu.ac.il

Abstract. Let a text string T of n symbols and a pattern string P of m symbols from alphabet Σ be given. A *swapped version* P' of P is a length m string derived from P by a series of *local swaps*, (i.e. $p'_\ell \leftarrow p_{\ell+1}$ and $p'_{\ell+1} \leftarrow p_\ell$) where each element can participate in *no more than one swap*. The *Pattern Matching with Swaps* problem is that of finding all locations i of T for which there exists a swapped version P' of P with an exact matching of P' in location i of T.
Recently, some efficient algorithms were developed for this problem. Their time complexity is better than the best known algorithms for pattern matching with mismatches. However, the *Approximate Pattern Matching with Swaps* problem was not known to be solved faster than the pattern matching with mismatches problem.
In the *Approximate Pattern Matching with Swaps* problem the output is, for every text location i where there is a swapped match of P, the *number* of swaps necessary to create the swapped version that matches location i. The fastest known method to-date is that of counting mismatches and dividing by two. The time complexity of this method is $O(n\sqrt{m \log m})$ for a general alphabet Σ.
In this paper we show an algorithm that counts the number of swaps at every location where there is a swapped matching in time $O(n \log m \log \sigma)$, where $\sigma = min(m, |\Sigma|)$. Consequently, the total time for solving the approximate pattern matching with swaps problem is $O(f(n,m) + n \log m \log \sigma)$, where $f(n,m)$ is the time necessary for solving the pattern matching with swaps problem.

Key Words: Design and analysis of algorithms, combinatorial algorithms on words, pattern matching, pattern matching with swaps, non-standard pattern matching, approximate pattern matching.

1 Introduction

The *Pattern Matching with Swaps* problem (the *Swap Matching* problem, for short) requires finding all occurrences of a pattern of length m in a text of length n. The pattern is said to match the text at a given location i if adjacent pattern characters can be swapped, if necessary, so as to make the pattern identical to the substring of the text starting at location i. All the swaps are constrained to be disjoint, i.e., each character is involved in at most one swap.

The importance of the swap matching problem lies in recent efforts to understand the complexity of various *generalized pattern matching* problems. The textbook problem of *exact string matching* that was first shown to be solvable in linear time by Knuth, Morris and Pratt [10] does not answer the growing requirements stemming from advances in Multimedia, Digital libraries and Computational Biology. To this end, pattern matching has to adapt itself to increasingly broader definitions of "matching" [18, 17]. In computational biology one may be interested in finding a "close" mutation, in communications one may want to adjust for transmission noise, in texts it may be desirable to allow common typing errors. In multimedia one may want to adjust for lossy compressions, occlusions, scaling, affine transformations or dimension loss.

The above applications motivated research of two new types – *Generalized Pattern Matching*, and *Approximate Pattern Matching*. In generalized matching the input is still a text and pattern but the "matching" relation is defined differently. The output is all locations in the text where the pattern "matches" under the new definition of match. The different applications define the matching relation. An early generalized matching was the *string matching with don't cares* problem defined by Fischer and Paterson [8]. Another example of a generalized matching problem is the *less-than matching* [4] problem defined by Amir and Farach. In this problem both text and pattern are numbers. One seeks all text locations where every pattern number is less than its corresponding text number. Amir and Farach showed that the less-than-matching problem can be solved in time $O(n\sqrt{m}\log m)$.

Muthukrishnan and Ramesh [15] prove that practically all general matching relations, where the generalization is in the definition of single symbol matches, are equivalent to the boolean convolutions, i.e. it is unlikely that they could be solved in time faster than $O(n\log m)$, where n is the text length and m is the pattern length. As we have seen, some examples have significantly worse upper bound than this.

The swap matching problem is also a generalized matching problem. It arises from one of the edit operations considered by Lowrance and Wagner [14, 19] to define a distance metric between strings.

Amir et al [3] obtained the first non-trivial results for this problem. They showed how to solve the problem in time $O(nm^{1/3}\log m\log\sigma)$, where $\sigma = \min(|\Sigma|, m)$. Amir et al. [5] also give certain special cases for which $O(m\text{polylog}(m))$ time can