Theoretical Computer Science 525 (2014) 23–29

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Modulated string searching ☆

Alberto Apostolico [a,b,1], Péter L. Erdős [c,*,2], István Miklós [c,3], Johannes Siemons [d]

[a] College of Computing, Georgia Institute of Technology, 801 Atlantic Drive, Atlanta, GA 30318, USA
[b] Istituto di Analisi dei Sistemi e Informatica, Consiglio Nazionale delle Ricerche, Viale Manzoni 30, Roma, Italy
[c] Alfréd Rényi Institute of Mathematics, Reáltanoda u 13-15, Budapest 1053, Hungary
[d] School of Mathematics, University of East Anglia, Norwich, UK

ARTICLE INFO

Keywords:
Pattern matching with character classes
Karatsuba's fast multiplication algorithm
Locally bounded L_1-norm string matching on character classes
Truncated L_1-norm string matching on character classes

ABSTRACT

In his 1987 paper entitled *Generalized String Matching* Abrahamson introduced the concept of *pattern matching with character classes* and provided the first efficient algorithm to solve this problem. The best known solution to date is due to Linhart and Shamir (2009).
Another broad yet comparatively less intensively studied class of string matching problems is numerical string searching, such as for instance "less-than" or L_1-norm string searching. The best known solutions for problems in this class are based on FFT convolution after some suitable re-encoding.
The present paper introduces *modulated string searching* as a unified framework for string matching problems where the numerical conditions can be combined with some Boolean/numerical decision conditions on the character classes. One example problem in this class is the *locally bounded* L_1*-norm* matching problem with parameters b and τ: here the pattern "matches" a text of same length if their L_1-distance is at most b and if furthermore there is no position where the text element and pattern element differ by more than the local bound τ. A more general setup is that where the pattern positions contain character classes and/or each position has its own private local bound. While the first variant can clearly be handled by adaptation of the classic FFT method, the second one is far too complicated for this treatment. The algorithm we propose in this paper can solve all such problems efficiently.
The proposed framework contains two nested procedures. The first one, based on Karatsuba's fast multiplication algorithm, solves pattern matching with character classes within time $O(nm^{0.585})$, where n and m are the text and pattern length respectively (under some reasonable conventions). This is slightly better than the complexity of Abrahamson's algorithm for generalized string matching but worse than algorithms based on FFT. The second procedure, which works as a plug-in within the first one and is tailored to the specific problem variant at hand, solves the numerical and/or Boolean matching problem with high efficiency. Some of the previously known constructions can be adapted to match or outperform several (but not all) problem variations handled by the construction

☆ This research was carried out in part while A. Apostolico and J. Siemons were visiting the Rényi Institute, with support from the Hungarian Bioinformatics MTKD-CT-2006-042794 and Marie Curie Host Fellowships for Transfer of Knowledge.
* Corresponding author.
E-mail addresses: axa@cc.gatech.edu (A. Apostolico), erdos.peter@renyi.mta.hu (P.L. Erdős), miklos.istvan@renyi.mta.hu (I. Miklós), j.siemons@uea.ac.uk (J. Siemons).
[1] Additional support was provided by the United States–Israel Binational Science Foundation (BSF) Grant No. 2008217 and by the Research Program of Georgia Tech.
[2] Research supported in part by the Hungarian NSF under contracts NK 78439 and K 68262.
[3] Research supported in part by the Hungarian NSF under contract PD 84297.

0304-3975/$ – see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.tcs.2013.10.013

proposed here. The latter aims to be a general tool that provides a unified solution for all problems of this kind.

© 2013 Elsevier B.V. All rights reserved.

1. Introduction

String searching is a basic primitive of computation. In the standard formulation of the problem, we are given a pattern and a text and are required to find all occurrences of the pattern in the text. Several variants of the problem have also been considered, such as allowing mismatches, insertions, deletions, swaps and so on.

In his paper [1] Abrahamson introduced the notion of *pattern matching with character classes* (or **PMCC** for short) which is specified as follows. The *pattern* P of length m is given as a sequence of character classes ($P[j] \subseteq \Sigma$) and the *text* T is a sequence from Σ^* (that is $T[i] \in \Sigma$). Here P occurs at location i in T if $\forall j\colon\ 1 \leqslant j \leqslant m,\ \ T[i+j-1] \in P[j]$. The problem of PMCC for a (typically long) text is to find all positions in the text T where the pattern P occurs. Standard string searching thus corresponds to the special case where each character class consists of exactly one element. In the original formulation PMCC was called *generalized pattern matching*.

Abrahamson proved that PMCC is harder than standard string searching and gave an algorithm for it. Since the algorithm deals with unrestricted alphabets, both the text and the pattern are encoded over the fixed auxiliary alphabet $\phi, 0, 1$. More precisely the text alphabet Σ is presumed to be the infinite set $\{\phi, a_l, a_2, \ldots\}$, where a_i is represented by the string $\#\bar{\imath}$, where $\bar{\imath}$ is the binary representation of i, without leading zeros. Symbol ϕ is represented by itself.

Now, let $\hat{M}$ denote the number of symbols used over the original alphabet to describe the pattern elements, and let M be the total length of the encoding of the pattern. Likewise, let n be the number of symbols in the text sequence, and N the total length of the encoding of the text. Then the time complexity of Abrahamson's algorithm is

$$O\left(M + N + n\hat{M}^{1/2}\,\mathrm{polylog}(m)\right).$$

The state of the art for PMCC is due to Linhart and Shamir [7]. Their algorithm has the following impressive time complexity: having set $\kappa = \log_{|\Sigma|}(\log n/\log m)$, then it is $O(|\Sigma|^{1-\kappa} n \log m)$ for $\kappa \leqslant 1$, while for $\kappa > 1$ it becomes $O(n \log(m/\kappa))$. Their approach can be extended to solve PMCC with mismatches and to PMCC with subset matching. It is based on encoding the text and pattern using large prime numbers, and on an FFT-based convolution process. It is suitable for checking "element(s) in a subset relation" but not for more complicated conditions.

The problem of searching for strings consisting of numerical values rather than characters arises in countless applications and some variants have already been studied in combinatorial pattern matching. In these problems the *fitting* conditions are described in numerical terms. For example, in the *less-than* string searching problem (Amir and Farach [2]), the pattern fits the text if at each position of the alignment the pattern value does not exceed the corresponding text value. Additional variants require the computation of the L_1-*distance* of the pattern from the text at each starting position (Amir, Landau and Vishkin [4], Lipsky [8]). Yet another version, known as the k–L_1-*distance problem* (Amir, Lipsky, Porat and Umanski [3]), consists of computing approximate matching in the L_1-metric.

These fast methods are also based on suitable encoding processes and on FFT, with corresponding time complexity. These algorithms do not seem to be applicable to numerical string searching with character classes and in general to those cases where a pointwise evaluation of individual comparisons is required.

In the next section we introduce the *modulated string searching* framework (or **MSS** for short) which combines the flexibility of PMCC with numerical calculations and/or more complicated Boolean conditions. We will give first a simple and naïve solution for the problem. (See Section 2.)

Our proposed approach for MSS is by a pair of nested procedures. The first one (see Section 3) is an algorithm to solve PMCC, based on Karatsuba's fast multiplication method. Its complexity is

$$O\left(nm^{0.585}\right)$$

where n and m are the text and pattern lengths, respectively, provided that all other parameters involved such as character class number, etc., can be treated as constants. Here one could argue that the application of the Toom–Cook or the Schönhage algorithms [5,9] yields a better performance. This is true, however, only for certain values of the text and pattern lengths. In addition, those algorithms require higher overheads, offsetting the overall gain. The above complexity is also worse than the complexity achieved in, say, [7]. However, the present method allows us to design a second procedure which works as a plug-in within the first one (see Section 4) and which solves a variety of numerical and/or Boolean problems. Indeed, the first procedure of our framework is always the same, while the plug-in procedure and its complexity depend heavily on specific matching conventions. Some of the previously known constructions can be adapted to match or outperform several (but not all) problem variations handled by the method proposed here, which therefore aims to be a general tool that provides a unified solution for all problems of this kind.

Download English Version:

https://daneshyari.com/en/article/434349

Download Persian Version:

https://daneshyari.com/article/434349

Daneshyari.com