Theoretical Computer Science ••• (••••) •••–•••

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Covering problems for partial words and for indeterminate strings

Maxime Crochemore [a,b], Costas S. Iliopoulos [a], Tomasz Kociumaka [c], Jakub Radoszewski [a,c,*], Wojciech Rytter [c], Tomasz Waleń [c]

[a] Department of Informatics, King's College London, UK
[b] Université Paris-Est, France
[c] Institute of Informatics, University of Warsaw, Poland

ARTICLE INFO

Article history:
Received 24 February 2017
Received in revised form 29 April 2017
Accepted 7 May 2017
Available online xxxx

Keywords:
Cover of a word
Partial word
String with don't cares
Indeterminate string
Fixed-parameter tractability

ABSTRACT

Indeterminate strings are a subclass of non-standard words having non-deterministic nature. In a classic string every position contains exactly one symbol—we say it is a *solid* symbol—while in an indeterminate string a position may contain a set of symbols (possible at this position); such sets are called *non-solid* symbols. The most important subclass of indeterminate strings are partial words, where each non-solid symbol is the whole alphabet; in this case non-solid symbols are also called *don't care* symbols. We consider the problem of finding a *shortest cover* of an indeterminate string, i.e., finding a shortest *solid* string whose occurrences cover the whole indeterminate string. We show that this classical problem becomes NP-complete for indeterminate strings and even for partial words. The proof of this fact is one of the main results of this paper. Our other main results focus on design of algorithms efficient with respect to certain parameters of the input (so called FPT algorithms) for the shortest cover problem. For the indeterminate string covering problem we obtain an $O(nk^2 + 2^k k^3)$-time algorithm, where k is the number of non-solid symbols, while for the partial word covering problem we obtain a running time of $O(nk^2 + 2^{O(\sqrt{k}\log k)})$. Additionally, we prove that, unless the Exponential Time Hypothesis is false, no $2^{o(\sqrt{k})}n^{O(1)}$-time solution exists for either problem, which shows that our algorithm for partial words is close to optimal. We also present an algorithm for both problems parameterized both by k and the alphabet size with a simple implementation.
A preliminary version of this article was presented at the 25th International Symposium on Algorithms and Computation (ISAAC 2014), LNCS, vol. 8889, pp. 220–232, Springer (2014) [12].

1. Introduction

A classic string is a sequence of symbols from a given alphabet Σ. In an *indeterminate* string, some positions may contain, instead of a single symbol from Σ (called a *solid* symbol), a non-empty subset of Σ. Such a *non-solid* symbol can mean that

* Corresponding author.
E-mail addresses: maxime.crochemore@kcl.ac.uk (M. Crochemore), c.iliopoulos@kcl.ac.uk (C.S. Iliopoulos), kociumaka@mimuw.edu.pl (T. Kociumaka), jrad@mimuw.edu.pl (J. Radoszewski), rytter@mimuw.edu.pl (W. Rytter), walen@mimuw.edu.pl (T. Waleń).

http://dx.doi.org/10.1016/j.tcs.2017.05.026
0304-3975/

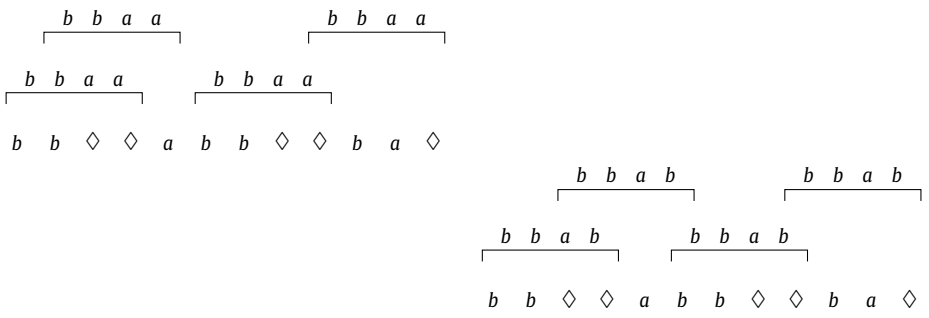

Fig. 1. Partial word $bb\Diamond\Diamond abb\Diamond\Diamond ba\Diamond = bb\{a,b\}\{a,b\}abb\{a,b\}\{a,b\}ba\{a,b\}$ with its two shortest covers.

the exact symbol at the given position is not known, but is suspected to be one of the specified symbols. The simplest type of indeterminate strings are *partial words*, in which every non-solid symbol is a don't care symbol, denoted here $\Diamond$ (other popular notation is $*$), which represents the whole alphabet Σ.

Motivations for indeterminate strings can be found in computational biology, musicology, and other areas. In computational biology, analogous juxtapositions may count as matches in protein sequences. In fact, the FASTA format[1] representing nucleotide or peptide sequences specifically includes indeterminate letters. In music, single notes may match chords, or notes separated by an octave may match; see [16].

Algorithmic study of indeterminate strings is mainly devoted to pattern matching. The first efficient algorithm was proposed by Fischer and Paterson for strings with don't care symbols [13]. Faster algorithms for this case were afterwards given in [29,22,23,10,9]. Pattern matching for general indeterminate strings, known as generalized string matching, was first considered by Abrahamson [1] in the variant that the text is solid. In the most general variant, pattern matching on indeterminate strings probably cannot be solved efficiently in general [19]. There were practical approaches to the problem; see [16,30] for some recent examples. A survey on partial words, related mostly to their combinatorics, can be found in a book by Blanchet-Sadri [7].

The notion of *cover* belongs to the area of quasiperiodicity, that is, a generalization of periodicity in which the occurrences of the period may overlap [4]. A cover of a solid string S is a string that covers all positions of S with its occurrences. Covers in solid strings were already extensively studied. A linear-time algorithm finding the shortest cover of a string was given by Apostolico et al. [5] and later on improved into an on-line algorithm by Breslauer [8]. A linear-time algorithm computing all the covers of a string was proposed by Moore and Smyth [28]. Afterwards an on-line algorithm computing all the covers of a string was given by Li and Smyth [26].

Other types of quasiperiodicities are seeds [18,24] and numerous variants of covers and seeds, including approximate and partial covers and seeds.

Classical periodicities, including periods and the border array, were already extensively studied in the field of indeterminate strings (originating from the papers [15,17]) and also for partial words (see, e.g., the book of Blanchet-Sadri [7]). Quasiperiodicity is a relaxation of the classical notions of periodicity and, as such, enables one to find approximate repetitive structures. This is important in imprecise data that occurs often in such areas of applications as computational biology and music.

The problem that we consider in this work is as follows (see also Fig. 1):

> COVERING AN INDETERMINATE STRING
> **Input:** an indeterminate string T
> **Output:** the length of a shortest solid cover of T

We allow the same non-solid symbol to match two different solid symbols for two different occurrences of the same cover. All our algorithms can actually recover an example shortest cover.

Throughout the paper we use the following notations: n is the length of the given indeterminate string T, k is the number of non-solid symbols in T, and σ is the size of the underlying alphabet Σ. We assume that $2 \le \sigma \le n$ with $\Sigma = \{1, \ldots, \sigma\}$ and that each non-solid symbol in the indeterminate string is represented by a bit vector of size σ. Thus the size of the input is $O(n + \sigma k)$.

The first attempts to the problem of indeterminate string covering were made in [3,6,17]. The common assumption of these papers is that $\sigma = O(1)$. The earliest contribution [17] does not provide a rigorous definition of a cover of an

[1] http://en.wikipedia.org/wiki/FASTA_format.

Please cite this article in press as: M. Crochemore et al., Covering problems for partial words and for indeterminate strings, Theoret. Comput. Sci. (2017), http://dx.doi.org/10.1016/j.tcs.2017.05.026