Journal of Discrete Algorithms 32 (2015) 24–43

Contents lists available at ScienceDirect

Journal of Discrete Algorithms

www.elsevier.com/locate/jda

The complexity of string partitioning ☆

Anne Condon [a], Ján Maňuch [a,b,*,1], Chris Thachuk [c,2]

[a] Dept. of Computer Science, University of British Columbia, Vancouver BC, Canada
[b] Dept. of Mathematics, Simon Fraser University, Burnaby BC, Canada
[c] Dept. Computer Science, University of Oxford, Oxford, UK

ARTICLE INFO

Article history:
Available online 4 December 2014

Keywords:
String partitioning
Equality-free
Prefix-free
Suffix-free
Factor-free
NP-completeness
Collision-aware oligo design
Gene synthesis

ABSTRACT

Given a string w over a finite alphabet Σ and an integer K, can w be partitioned into strings of length at most K, such that there are no *collisions*? We refer to this question as the *string partition* problem and show it is **NP**-complete for various definitions of collision and for a number of interesting restrictions including $|\Sigma| = 2$. This establishes the hardness of an important problem in contemporary synthetic biology, namely, oligo design for gene synthesis.

1. Introduction

Many problems in genomics have been solved by the application of elegant polynomial-time string algorithms, while others amount to solving known **NP**-complete problems; for instance, sequence assembly amounts to solving *shortest common superstring* [13], and genome rearrangement to *sorting strings by reversals and transpositions* [3]. The hardness of these problems has motivated extensive research into heuristic algorithms as well as polynomial-time algorithms for useful restrictions [8,10–12,16,18,21]. In a similar vein, we establish the hardness of the following fundamental question: can a string be partitioned into factors (i.e. substrings), of bounded length, such that no two *collide*? We refer to this as the *string partition* problem and study it under various restrictions and definitions of what it means for two factors to *collide*.

The study of string partitioning is motivated by an increasingly important problem arising in contemporary synthetic biology, namely gene synthesis. This technology is emerging as an important tool for a number of purposes including the determination of RNAi targeting specificity of a particular gene [14], design of novel proteins [7] and the construction of complete bacterial genomes [9]. There have been numerous studies utilizing synthetic genes to determine the potential of gene vaccines [2,4,15,19]. Despite the tremendous need for synthetic genes for both interrogative studies and for therapeutics, construction of genes, or any long DNA or RNA sequence, is not a trivial matter. Current technology can only produce short oligonucleotides (oligos) accurately. As such, a common approach is to design a set of oligos — short sequences of nucleic acids — that could assemble into the desired sequence [20].

☆ Authors are listed in the alphabetic order.

* Corresponding author.
E-mail addresses: condon@cs.ubc.ca (A. Condon), jmanuch@cs.ubc.ca (J. Maňuch), chris.thachuk@cs.ox.ac.uk (C. Thachuk).

[1] Research supported by NSERC Discovery Grant No. 371978-2010.
[2] Research supported by ERC Advanced Grant VERIWARE EPSRC Grant EP/G037930/1 and Oxford Martin School.

http://dx.doi.org/10.1016/j.jda.2014.11.002
1570-8667/

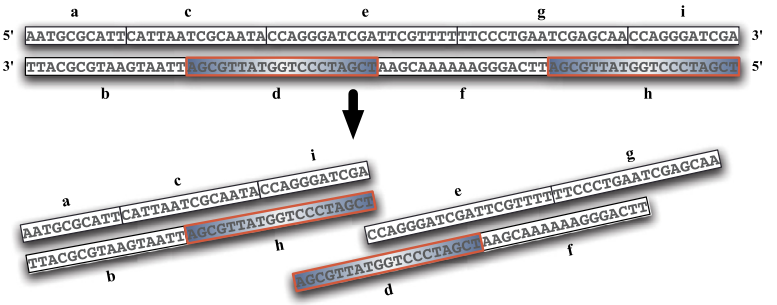

Fig. 1. An intended self-assembly (top) of a set of oligos — short strands of DNA, labeled a through h — for a desired DNA duplex. A foiled self-assembly (bottom) of the same oligos due to d and h being identical.

Fig. 2. (Left) Two partitions are shown for the string *mississippi*. The selected strings in both partitions have maximum length 2. The partition shown above the string is factor-free: no selected string is a factor of another; however, the partition shown below the string is not factor-free. (Right) A valid factor-free multiple string partition of a set of three strings into selected strings of maximum length 3.

To understand the connection between string partitioning and gene synthesis, consider the following. A DNA *oligo*, or *strand* is a string over the four letter alphabet $\{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{T}\}$. The *reverse complement* F' of an oligo F is determined from F by replacing each $\mathtt{A}$ with a $\mathtt{T}$ and vice versa, each $\mathtt{C}$ with a $\mathtt{G}$ and vice versa, and reversing the resulting string. Two DNA oligos F and F' are said to *hybridize* if a sufficiently long factor of F is the reverse complement of a factor of F' (see Fig. 1). A DNA *duplex* consists of a positive strand and its reverse complement, the negative strand. The *collision-aware oligo design for gene synthesis* (CA-ODGS) problem is to determine cut points in the positive and negative strands, which demarcate the oligos to be synthesized, such that the resulting design will successfully self-assemble. For the oligos to self-assemble correctly, they should 1) alternate between the positive and negative strands, with some overlap between successive oligos, and 2) only hybridize to the oligos they overlap with by design. Since there is variability in the length of the selected oligos, there are exponentially many designs.

In previous work [5], the authors provided some evidence that the CA-ODGS problem may be hard by showing that partitioning a string into factors, of bounded length, such that no two are equal is **NP**-complete, even for strings over a quaternary alphabet. See Fig. 1 for an example design that assembles incorrectly into two fragments, with the wrong ordering of oligos and therefore primary sequence, due to identical oligos. In this work, we study the underlying string partition problem in much greater detail. We show that partitioning strings such that no selected string is a copy/factor/prefix&suffix/prefix/suffix of another is **NP**-complete. We begin by showing that the more general problem of partitioning a set of strings is hard and then we show how those instances can be reduced to single string instances, for each respective definition of collision. See Fig. 2 for an example of a single string instance (left) and set of strings instance (right). In all cases, we demonstrate that the problems remain hard even when restricted to binary strings. A preliminary version of this work with the proofs for the equality-free case has appeared in [6].

2. Preliminaries

A *string* w is a sequence of letters over an alphabet Σ. Let $|w|$ denote the length of w, w^{R} a mirror image (reversal) of w, and let $(w)^i$ denote the string w repeated i times. The empty string is denoted as ε. String x is a *factor* of w if $w = \alpha x \beta$, for some (possibly empty) strings α and β. Similarly, x is a *prefix* (*suffix*) of w if $w = x\beta$ ($w = \alpha x$) for some (possibly empty) strings α and β. The prefix (suffix) of length k of w will be denoted as $\mathrm{prefix}_k(w)$ ($\mathrm{suffix}_k(w)$). Similarly, let $\mathrm{factor}_{i,j}(w)$ denote the factor of w of length $j - i$ starting at position i. Note that $\mathrm{prefix}_i(w) = \mathrm{factor}_{1,i+1}(w)$.

A *K-partition* of w is a sequence $P = p_1, p_2, \ldots, p_l$, for some l, where each p_i is a string over Σ of length at most K and $w = p_1 p_2 \ldots p_l$. We say that strings $p_1, \ldots, p_l$ are *selected* in the K-partition and that strings $p_i \ldots p_j$, $1 \le i \le j \le l$, are *super-selected*, with respect to the selected strings. We say P is *equality-free*, *prefix-free*, *suffix-free*, *prefix&suffix-free* or *factor-free* if for all i, j, $1 \le i \ne j \le l$, neither p_i nor p_j is a copy, prefix, suffix, prefix or suffix, or factor, respectively, of the other. We say such partitions are *valid* (for the problem in question); otherwise, we say the partition contains a *collision*. We generalize the notion of a K-partition to a set of strings $\mathcal{W}$ to mean a K-partition for each string in $\mathcal{W}$. The length of $\mathcal{W}$ is the combined length of the strings in the set and will be denoted by $\|\mathcal{W}\|$. A K-partition for a set of strings is valid if no two elements in any, possibly different, partition collide. Finally, we will refer to the boundaries of a partition of string w as *cut points*, where the first cut point 0 and the last cut point $|w|$ are called trivial. For instance, the first partition of mississippi in Fig. 2 has the following non-trivial cut points $1, 3, 5, 7$ and 9. We say that a partition P is a *refinement* of