Computing the Minimum Approximate $\lambda$-Cover of a String

Qing Guo$^1$, Hui Zhang$^2$, and Costas S. Iliopoulos$^2$

$^1$Department of Computer Science and Engineering, Zhejiang University, Hangzhou, Zhejiang 310027, China
guoqing@tiansign.com
$^2$Department of Computer Science, King’s College London Strand, London WC2R 2LS, England
{hui, csi}@dcs.kcl.ac.uk

Abstract. This paper studies the minimum approximate $\lambda$-cover problem of a string. Given a string $x$ of length $n$ and an integer $\lambda$, the minimum approximate $\lambda$-cover problem is to find a set of $\lambda$ substrings of equal length that covers $x$ with the minimum error, under a variety of distance models including the Hamming distance, the edit distance and the weighted edit distance. We present an algorithm that can solve this problem in polynomial time.

1 Introduction

String regularities mainly concern the repetitive structures of strings. Typically, a substring $w$ is a period of a given string $x$ if $x$ is a prefix of a string constructed by concatenations of $w$. By allowing superpositions as well as concatenations, the notion of periods is generalized, called covers.

In the literature, a tremendous amount of research has been done on computing the covers of a given string $x$ of length $n$. Apostilico, Farach and Iliopoulos$^1$ first introduced the notion of covers and presented a linear-time algorithm to test superprimitivity. Breslauer$^2$ described a linear time on-line algorithm to compute the shortest cover of every prefix of $x$. Moore and Smyth$^8$, and recently Li and Smyth$^7$ both gave a solution to find all the covers of $x$. In parallel computation, Iliopoulos and Park$^5$ gave a work-time optimal $O(\log \log n)$ algorithm for the shortest cover problem of $x$.

Extending the definition of covers in the sense that a set of substrings instead of a single substring of $x$ are examined, Zhang et al.$^{11}$ introduced the notion of $\lambda$-covers. Given an integer $\lambda$, the $\lambda$-cover problem attempts to find all the sets of $\lambda$ substrings each of equal length that cover $x$. A general algorithm that can solve this problem in $O(n^2)$ time was also presented.

Advances in multimedia technology and computational biology has shown that it could be of significant benefit to relax the basis for regularities. For instance, one seldom expects to find exact repetitions in molecular sequence analysis, but approximate regularities that allow errors to some extent. In this case, we consider a string ”matching” a given pattern if the distance between them is...
within allowed bounds under a predefined metric. The most classic metrics are *Hamming distance*, *edit distance* and *weighted edit distance*.

Sim, Iliopoulos, Park and Smyth [9] introduced the notion of *approximate period* and provided polynomial time algorithms for finding approximate periods. Then, Sim, Park, Kim and Lee [10] solved the *approximate covers* problem in polynomial time as well.

Inspired by the idea of $\lambda$-covers, we introduce in this paper the notion of *approximate $\lambda$-covers*, approximate version of $\lambda$-covers. The motivation comes from the need for biological sequence analysis and its interaction with the alignment problem as solved by BLAST. It could be considered as a preliminary operation before the alignment of two sequences because it discovers motifs having adequate properties for that. We focus on solving the minimum approximate $\lambda$-cover problem of a string. To avoid triviality, we simply consider the case $\lambda > 1$.

## 2 Preliminaries

A *string* is a sequence of zero or more symbols over an alphabet $\Sigma$. A string $x$ of length $n$ is represented by an array $x[1..n] = x[1]x[2] \cdots x[n]$, where $x[i]$ is the $i$-th symbol of $x$ ($x[i] \in \Sigma$ for $1 \leq i \leq n$). The *empty string* is the empty sequence (of zero length) denoted by $\varepsilon$. The set of all strings over the alphabet $\Sigma$ (including the empty string) is denoted by $\Sigma^*$. A string $w$ is a *substring* of $x$ if $x = uwv$ for $u, v \in \Sigma^*$; $x$ is equivalently a *superstring* of $w$. A substring of length $p$ is called a *$p$-substring* for short. For a nonempty substring $w = x[i..j]$, we say that $w$ occurs at position $i$ and $i$ is an occurrence of $w$ in $x$. A string $w$ is a *prefix* of $x$ if $x = wu$ for $u \in \Sigma^*$. Similarly, $w$ is a *suffix* of $x$ if $w = uw$ for $u \in \Sigma^*$.

The string $xy$ is a *concatenation* of two strings $x$ and $y$. The concatenation of $k$ copies of $x$ is denoted by $x^k$. For two strings $x = x[1..n]$ and $y = y[1..m]$ such that $x[n-i+1..n] = y[1..i]$ for some $i \geq 1$, the string $x[1..n]y[i+1..m]$ is a *superposition* of $x$ and $y$ with $i$ overlaps, we say that $x$ and $y$ are *overlapping*. A substring $u$ is said to be a *cover* of $x$ if $x$ can be constructed by concatenations and superpositions of $u$. For example, the string $x = abababa$ has a cover $aba$.

The distance $\delta(x, y)$ between two strings $x$ and $y$ indicates the minimum cost to transform $x$ into $y$. The cost arises from a sequence of operations, which is the sum of the cost of the individual operations. The most frequent operations that are used for string transformation mainly consist of:

- *Insertion*: inserting an extraneous character $a$, denoted by $\varepsilon \rightarrow a$.
- *Deletion*: deleting a character $a$, denoted by $a \rightarrow \varepsilon$.
- *Substitution*: Replacing a character $a$ by another character $b$ ($b \neq a$), denoted by $a \rightarrow b$.

The following commonly used distance functions rest on above operations:

- *Hamming distance*: allows only substitutions, costing 1 for each operation. Note that, this model is restricted to two strings of the same length.