An Approximation Algorithm for Alphabet Indexing Problem
(Extended Abstract)

Shinichi Shimozono

Department of Control Engineering and Science,
Kyushu Institute of Technology
Kawazu 680-4, Iizuka, 820 JAPAN
E-mail: s@ceskyutech.ac.jp

Abstract. A K-indexing is a mapping from an alphabet \( \Sigma \) to a set \( I \) of \( K \) symbols forming the homomorphism to transform strings. Given \( \Sigma \), two disjoint sets of strings \( P, Q \) over \( \Sigma \) and \( I = \{1, \ldots, K\} \), Alphabet Indexing is the problem to find a K-indexing that transforms no two different strings taken from each \( P \) and \( Q \) into the same one. Although this problem is NP-complete, applying K-indexing to input data brings remarkable advantages in actual applications. In this paper, we introduce Max K-Indexing, a maximization version of Alphabet Indexing, that intends to maximize the number of pairs in \( P \times Q \) whose strings are not transformed into the same ones. We show that the problem is MAX SNP-hard, that is, the problem seems to have no polynomial-time algorithm achieving an arbitrary small error ratio. Then we propose a simple polynomial-time greedy algorithm and show that the algorithm attains the constant error ratio \( 1/K \) for K-indexing. Also we define M-Tuple K-Indexing problem by extending pairs of strings in Max K-Indexing to tuples of strings over more than two sets, and show that a natural extension of the algorithm also achieves a constant error bound.

1 Introduction

In actual applications of combinatorial problems of strings, the size of the alphabet of input data has strong influence on the efficiency of algorithms even it is a constant factor in a theoretical analysis. In bioinformatical knowledge acquisition, Arikawa et al. showed in [1] that the use of a mapping reducing the size of the alphabet can improve efficiency and accuracy of machine learning algorithms. According to biochemical knowledge, they defined a mapping from 20 kinds of amino acid symbols to three categories, and transformed amino acid sequences used as training examples of the machine learning system. The mapping distinguished the transformed positive examples from negative ones, and they obtained successful experimental results for a certain protein classification problem.

From the view point of computer science, there are researches of "mechanical method" for finding such mappings, namely, alphabet indexing [10, 11]. Given
an alphabet $\Sigma$, disjoint sets of strings $P, Q \subseteq \Sigma^*$ and an indexing alphabet $I' = \{1, \ldots, K\}$, an alphabet indexing is a mapping $f$ from $\Sigma$ to $I'$ such that the transformation of strings derived from $f$ converts no two strings $p \in P$ and $q \in Q$ into identical ones. Unfortunately, the problem for finding an alphabet indexing is NP-complete [10]. Therefore we introduced the optimization problem to find an indexing that allows some overlaps between the transformed sets of strings, and defined the optimization problem that employs the machine learning algorithm in [1]. We designed a local search algorithm for the problem, showed the PLS-completeness [10], and obtained the successful empirical results [11]. However, this optimization problem uses a hypothesis produced by a machine learning algorithm to measure the goodness of indexings, and the local search algorithm has no proved error estimations. To take advantages of an alphabet indexing, we need a simple optimization problem of alphabet indexing and a polynomial-time approximation algorithm whose error can be estimated.

In this paper, we introduce an optimization problem MAX $K$-INDEXING. This problem requires to find a $K$-indexing $f : \Sigma \rightarrow \{1, \ldots, K\}$ that maximizes the number of the pairs in $P \times Q$ whose strings are transformed into different ones by $f$. We show the MAX SNP-hardness of the problem in the context of class MAX SNP [2, 3, 8, 9]. This asserts that the problem seems to have no polynomial-time approximation scheme, that is, there is a certain constant bound for error ratios of approximation algorithms that run in polynomial time. Then we present a simple greedy algorithm inspired by the well-known greedy algorithm for MAXIMUM SATISFIABILITY by Johnson [7], and show that our algorithm achieves a constant relative error $1/K$ for $K$-indexing, with providing the worst case instance. By this new approach to the alphabet indexing problem, we can find an appropriate indexing in reasonable time not only for knowledge acquisition problems but also for the lossy text compression processed by the character-based compression [4].

Additionally, we define the problem $M$-TUPLE $K$-INDEXING by extending pairs in MAX $K$-INDEXING to tuples of strings. Then we show that the polynomial time algorithm extended from the greedy algorithm for MAX $K$-INDEXING also achieves a constant error ratio.

2 Notations

Let $\Sigma$ and $\Gamma$ be finite alphabets, and let $s$ and $t$ be strings in $\Sigma^*$. The $i$th symbol of $s$ is denoted by $s[i]$. We say that $s$ and $t$ are the same if $s[i] = t[i]$ and $|s| = |t|$ for all $1 \leq i \leq |s|$, where $|s|$ denotes the length of $s$. We call a mapping $f$ from $\Sigma$ to $\Gamma$ an alphabet indexing of $\Sigma$ by $\Gamma$. Especially, an alphabet indexing that maps to $\Gamma' = \{1, \ldots, K\}$ is called a $K$-indexing. For $a \in \Sigma$, we say that a symbol $f(a) \in \Gamma$ is an index of $a$. The homomorphism $f^* : \Sigma^* \rightarrow \Gamma^*$ derived from $f$ is the transformation of strings from $s$ in $\Sigma^*$ to $f^*(s) = f(s[1]) \cdots f(s[|s|])$ in $\Gamma^*$. We say that a $K$-indexing $f$ distinguishes $s, t \in \Sigma^*$ if $f^*(s) \not= f^*(t)$. For convenience, we regard the range of $K$-indexings as $\{0, 1, \ldots, K\}$ when we talk