Complexity of Finding Alphabet Indexing

Shimozono, Shinichi
Department of Control Engineering and Science, Kyushu Institute of Technology

Miyano, Satoru
Research Institute of Fundamental Information Science, Kyushu University

出版情報：RIFIS Technical Report. 61, 1992-08. 九州大学理学部附属基礎情報学研究施設
バージョン：
権利関係：

Complexity of Finding Alphabet Indexing

Shinichi Shimozono
Department of Control Engineering and Science,
Kyushu Institute of Technology, Iizuka 820, Japan
e mail: sin@ces.kyutech.ac.jp

Satoru Miyano
Research Institute of Fundamental Information Science,
Kyushu University 33, Fukuoka 812, Japan
e mail: miyano@rifis.sci.kyushu-u.ac.jp

Abstract

For two finite disjoint sets P and Q of strings over an alphabet Σ, an alphabet indexing ψ for P, Q by an indexing alphabet Γ with $|\Gamma| < |\Sigma|$ is a mapping $\psi : \Sigma \to \Gamma$ satisfying $\tilde{\psi}(P) \cap \tilde{\psi}(Q) = \emptyset$, where $\tilde{\psi} : \Sigma^* \to \Gamma^*$ is the homomorphism derived from ψ. We defined this notion through experiments of knowledge acquisition from amino acid sequences of proteins by learning algorithms. This paper analyzes the complexity of finding an alphabet indexing. We first show that the problem is NP-complete. Then we give a local search algorithm for this problem and show a result on PLS-completeness.

Key words: Algorithm and Computational Complexity, Alphabet Indexing, Local Search Algorithm, NP-Complete, PLS-Complete

1 Introduction

Machine learning methods have been developed in [1, 2] to discover bioinformatical knowledge from amino acid sequences of proteins which are compiled together with their functional information in databases such as PIR [13]. The learning algorithm in [1] uses elementary formal systems [3] as the representation of concepts and the learning algorithm in [2] produces decision trees over regular patterns as hypotheses.

Both learning algorithms assume two sets of strings called the set of positive examples and the set of negative examples, and find a hypothesis which explains the given positive and negative examples.

For the problem of identifying transmembrane domains of proteins [9, 18], the set of positive examples consists of amino acid sequences describing the transmembrane domains of membrane proteins while the set of negative examples consists of sequences taken from non-transmembrane regions of proteins. In [1, 2], 20 symbols of amino acid residues are transformed to three symbols according to the hydropathy index of Kyte and Doolittle [9], which assigns a real number between -4.5 and 4.5 to each amino acid residue. It is known that this hydropathy index plays an essential role in transmembrane domain identification [9]. It has been noticed in [1, 2] that this transformation makes the learning algorithms more efficient and the produced hypotheses more accurate. Surprisingly, after transforming the positive and negative examples to strings over three symbols, the resulting sets of positive and negative examples do not have any large overlaps. Namely, the positiveness and negativeness of examples are kept under this transformation.

Inspired by this observation, we define a notion of an *alphabet indexing* and consider the problem of finding such an alphabet indexing. Formally, let P and Q be disjoint sets of strings over an alphabet Σ and let Γ be another alphabet with $|\Gamma| < |\Sigma|$. An alphabet indexing ψ for P, Q by Γ is a mapping from Σ to Γ such that $\tilde{\psi}(P)$ and $\tilde{\psi}(Q)$ do not have any overlaps, where $\tilde{\psi}(P)$ and $\tilde{\psi}(Q)$ are the sets of strings obtained from strings in P and Q by transforming each symbol by the mapping ψ. The alphabet Γ is called the indexing alphabet. Since the alphabet indexing reduces hypothesis spaces, it is important to develop efficient algorithms for finding alphabet indexings in knowledge acquisition from large amounts of data.

This paper establishes fundamental results on complexity issues of the alphabet indexing problem. First we prove that the problem of finding an alphabet indexing by an alphabet of a fixed size is computationally hard. There are two ways of coping with this hardness. One is to allow the indexing alphabet to be larger than the given one. The other is to allow overlaps between the transformed positive and negative examples. In this paper, we focus in the latter. Section 3 introduces a pseudo-indexing, which allows some overlaps between the transformed sets of strings. We regard this problem as a combinatorial optimization problem:

(1) Any pseudo-indexing is a feasible solution.

(2) The performance measure of an indexing is given by the accuracy of a hypothesis produced by the learning algorithm for the transformed inputs.

Then we give a local search strategy for searching a pseudo-indexing. We analyze its computational complexity according to the formulation of the class PLS defined by Johnson et al. [5] and show the PLS-completeness of a local search version of this problem. This strategy has been implemented in the system BONSAI [16] and the experiment in [16] has shown very successful results for the transmembrane domain identification problem and the signal peptide identification problem [17].

2 Indexing is NP-Complete

Let Σ and Γ be finite alphabets, and let ψ be a mapping from Σ to Γ. Then $\tilde{\psi}(s)$ for $s \in \Sigma^*$ and $\tilde{\psi}(S)$ for $S \subseteq \Sigma^*$ are defined by $\tilde{\psi}(s_1 \cdots s_n) = \psi(s_1) \cdots \psi(s_n)$ and $\tilde{\psi}(S) = \{\tilde{\psi}(s) \mid s \in S\}$. We denote $\tilde{\psi}(S)$ for $S \subseteq \Sigma^*$ by $\tilde{S}$ if there is no ambiguity. Let P and Q be disjoint sets of strings over Σ, and let Γ be an alphabet with $|\Gamma| < |\Sigma|$. Then we call a mapping ψ from Σ to Γ an *alphabet indexing of* Σ *by* Γ *for* P *and* Q if ψ satisfies $\tilde{P} \cap \tilde{Q} = \emptyset$ and the alphabet Γ is called its *indexing alphabet*. This condition preserves examples in the positive set P and the negative set Q being consistent by the transformation ψ.

Definition 1. Alphabet Indexing Problem (AIP).
Instance: A finite alphabet Σ, two disjoint finite sets of strings P, Q over Σ, and an indexing alphabet Γ with $|\Sigma| > |\Gamma|$.
Question: Is there an alphabet indexing of Σ by Γ for P and Q?

If the difference of the sizes $|\Sigma| - |\Gamma| = d$ is constant, this problem is solvable in $O(n^{2d})$ time. However, generally, the following results states that the problem of deciding whether the disjoint sets of strings have a consistent indexing or not is computationally hard, even for problems by two symbols.

Theorem 1. The alphabet indexing problem is NP-complete, even if the size of the indexing alphabet is two and the length of strings is bounded by three.

Proof. We give a reduction from NP-complete problem NOT-ALL-EQUAL 3SAT [4] (NAE 3SAT for short) to this problem. Given a pair of a set $X = \{x_1, \ldots, x_n\}$ of variables and a collection $C = \{c_1, \ldots, c_m\}$ of clauses over X such that each clause $c \in C$ has $|c| = 3$, NAE 3SAT is the problem to decide whether (X, C) is satisfiable, that is, there is a truth assignment for X such that each clause in C has at least one true literal and at least one false literal. Given a NAE 3SAT formula (X, C), we build an instance (Σ, Γ, P, Q) of the alphabet indexing problem as follows:

(i) $\Sigma = \{\mathtt{t}, \mathtt{f}\} \cup \{x_i, x_{\bar{i}} \mid 1 \leq i \leq n\}$, $\Gamma = \{0, 1\}$.

(ii) $P = \{\mathtt{tft}\} \cup \{x_i x_{\bar{i}} x_i \mid 1 \leq i \leq n\} \cup \{l_1 l_2 l_3 \mid (l_1, l_2, l_3) \in C\}$,
$Q = \{\mathtt{ttt}, \mathtt{fff}\}$.

Then (X, C) is satisfiable if and only if (Σ, Γ, P, N) has an alphabet indexing of Σ by Γ for P and Q.

if: If there exists an assignment $b = (b_1, \ldots, b_n) \in \{0, 1\}^n$ satisfying (X, C), we compute an indexing ψ by $\psi(x_i) = b_i, \psi(x_{\bar{i}}) = b_{\bar{i}}$ for all $1 \leq i \leq n$.

Since the assignment b satisfies (X, C), we have at least one 1-valued literal and at least one 0-valued literal for each clause. Thus, for any string corresponding to a clause, ψ maps at least one literal symbol to 1 and at least one literal symbol to 0. So, all strings $\tilde{\psi}(l_1 l_2 l_3)$ in $\tilde{\psi}(P)$ are distinct from neither $\tilde{\psi}(\mathtt{ttt})$ nor $\tilde{\psi}(\mathtt{fff})$ in $\tilde{\psi}(Q)$.

only if: If (Σ, Γ, P, N) has an indexing ψ, we compute an assignment b satisfying (X, C) by letting $b = \psi(x_1), \ldots, \psi(x_n)$.

Since ψ satisfies $\tilde{\psi}(\mathtt{tft}) \neq \tilde{\psi}(\mathtt{ttt})$, we have $\psi(\mathtt{t}) \neq \psi(\mathtt{f})$. Also, since ψ satisfies $\tilde{\psi}(x_i x_{\bar{i}} x_i) \neq \tilde{\psi}(\mathtt{ttt})$ and $\tilde{\psi}(x_i x_{\bar{i}} x_i) \neq \tilde{\psi}(\mathtt{fff})$, boolean value for x_i and $x_{\bar{i}}$ for all $1 \leq i \leq n$ satisfies $\neg x_i = x_{\bar{i}}$. Furthermore, since $\tilde{\psi}(l_1 l_2 l_3) \neq \tilde{\psi}(\mathtt{fff})$ and $\tilde{\psi}(l_1 l_2 l_3) \neq \tilde{\psi}(\mathtt{ttt})$ for any clause in C, at least one literal has the value same with $\psi(\mathtt{t})$ and at least one literal has the value same with $\psi(\mathtt{f})$. Thus any clause in (X, C) has at least one 1-valued literal and at least one 0-valued literal for the assignment b. □

The alphabet indexing problem considers whether there is a transformed string that comes from both P and Q. Therefore, in the proof of Theorem 1, strings of the same length are considered for finding an alphabet indexing. This result can be easily extended to the hardness of more general problem that considers substrings of the transformed strings.

Also, we can extend the above result to the alphabet indexing problem by the indexing alphabet Γ of the size $|\Gamma| \leq |\Sigma|/d$ for some constant $d > 1$. Without loss of generality, we assume that (X, C) is sufficiently large for satisfying $|X|+1 > d$. Given a NAE 3SAT formula (X, C), we build the same sets of strings P and Q as in the reduction in the proof of Theorem 1. In advance, we let $\Sigma = \{\mathtt{t}, \mathtt{f}, x_1, \overline{x}_1, \ldots, x_n, \overline{x}_n, y_1, \ldots, y_k\}$ and $\Gamma = \{0, 1, \ldots, k+1\}$, where $k = \left\lfloor \frac{2(n-d+1)}{d-1} \right\rfloor$. Finally, we add all of $\{y_i y_i y_i \mid 1 \leq i \leq k\}$ to P and $\{y_i y_j y_i \mid 1 \leq i \neq j \leq k\}$ to Q. Then any alphabet indexing ψ of this instance (Σ, Γ, P, Q) must satisfy $\psi(\mathtt{t}) \neq \psi(\mathtt{f})$, $\psi(x_i) \neq \psi(\overline{x}_i)$ for all $x_i \in X$ and also $\tilde{\psi}(l_1 l_2 l_3) \neq \tilde{\psi}(\mathtt{ttt})$, $\tilde{\psi}(l_1 l_2 l_3) \neq \tilde{\psi}(\mathtt{fff})$ for all $(l_1, l_2, l_3) \in C$ as in the proof of Theorem 1. Moreover, ψ must satisfy $\psi(y_i) \neq \psi(\mathtt{t})$ and $\psi(y_i) \neq \psi(\mathtt{f})$ for all $1 \leq i \leq k$, and $\psi(y_i) \neq \psi(y_j)$ for $1 \leq i \neq j \leq k$. This concludes the same result as in the case $|\Gamma| = 2$, and

$$\begin{aligned} |\Sigma|/|\Gamma| &= 1 + \frac{2n}{2+k} \\ &\geq 1 + \frac{2n}{2 + \frac{2(n-d+1)}{d-1}} = d, \end{aligned}$$

the inequality $|\Gamma| \leq |\Sigma|/d$ is satisfied by k.

The following result follows from Theorem 1 and these observations.

Collorary 1. Let P and Q be sets of strings over Σ, and let Γ be an alphabet with $2 \leq |\Gamma| \leq |\Sigma|/d$ for some constant $d > 1$. Let P' and Q' be the sets of strings defined as follows:

$$P' = \{s' \mid s' \text{ is a substring of } s \in P \text{ with } |s'| \geq 3\}$$
$$Q' = \{s' \mid s' \text{ is a substring of } s \in Q \text{ with } |s'| \geq 3\}$$

Then the problem of finding an alphabet indexing of Σ by Γ for $P' - Q'$ and $Q' - P'$ is NP-complete.

3 A local search strategy for finding an indexing

In this section, we introduce a pseudo-indexing, which allows the transformed sets to have nonempty intersection. Then we consider a local search algorithm for finding a pseudo-indexing.

Let Σ be an alphabet, and let (P, Q) be a pair of finite subsets $P, Q \subseteq \Sigma^*$ with weights (Ω^P, Ω^Q), where $\Omega^P : P \to \mathbf{Z}^+$ and $\Omega^Q : Q \to \mathbf{Z}^+$. Let A be an algorithm that takes a pair of sets of weighted strings $(P, Q; \Omega^P, \Omega^Q)$ as an input and produces a hypothesis $\langle h \rangle$ in the hypothesis space H_A representing a function $h : \Sigma^* \to \{0, 1\}$. We say that A is *robust* if, for any input, A produces a function h satisfying $P - Q \subseteq L_h$ and $Q - P \subseteq \overline{L_h}$, where $L_h = \{s \in \Sigma^* \mid h(s) = 1\}$ and $\overline{L_h} = \{s \in \Sigma^* \mid h(s) = 0\}$. We can use a hypothesis produced by the robust algorithm A as a classifier that completely separates P and Q except the part $P \cap Q$. For obtaining more desirable hypothesis from the algorithms, we require that A is not only robust but also maximizes the goodness value function $\mu(p_1, p_0, q_1, q_0)) = 1 - \frac{1}{|P|+|Q|}(I(p_1, q_1) + I(p_0, q_0))$, where

$$I(p, q) = \begin{cases} 0 & (\text{if } p = 0 \text{ or } q = 0) \\ -p \log \dfrac{p}{p+q} - q \log \dfrac{q}{p+q} & (\text{otherwise}), \end{cases}$$

and p_1 (resp. n_1) is the total weight of positive examples in P (resp. negative examples in Q) and L_h, i.e., $p_1 = \sum_{p \in P \cap L_h} \Omega^P(p)$, $n_1 = \sum_{q \in Q \cap L_h} \Omega^Q(q)$, and p_0 (resp. n_0) is the total weight of positive examples in P (resp. negative examples in Q) and in $\overline{L_h}$. The original form of this function is the information gain function hired in Quinlan's ID-3 system [14].

Let P' and Q' be small subsets of P and Q respectively. We can define a performance measure of a pseudo-indexing $\psi : \Sigma \to \Gamma$ by the accuracy of the hypothesis produced by $A(\tilde{\psi}(P'), \tilde{\psi}(Q'); \Omega^{P', \psi}, \Omega^{Q', \psi})$ over P and Q, where

$$\Omega^{S, \psi}(s) = \sum_{s' \in S \,\&\, \tilde{\psi}(s) = \tilde{\psi}(s')} \Omega^S(s').$$

From this point of view, we formulated the problem for finding a pseudo-indexing as a combinatorial optimization problem and applied a local search strategy to the knowledge acquisition system that searches a pseudo-indexing and a decision tree over regular patterns [16]. This algorithm starts from any pseudo-indexing and iterates the process of improving a pseudo-indexing by its small displacements, until no more improvement is achieved. Then it outputs a pseudo-indexing and a decision tree that classifies given strings. Informally, a decision tree is a rooted tree that consists of nodes with queries and leaves with class names. To classify an input, we evaluate their queries from the root to a leaf according to their answers. When we reach a leaf,

we classify the input according to the label on that leaf. To query nodes, Arikawa et al. [2] attached simple regular patterns. This algorithm is robust and employing the goodness function defined above.

Now we formulate a local search strategy for the knowledge acquisition system by the polynomial-time local search problem (PLS) [5, 7, 11, 15]. First we review the definitions for the class PLS.

Definition 2. Let Σ be a finite alphabet. A *polynomial time local search problem* L is either a maximization or minimization problem specified as follows:

(1) $I(L) \subseteq \Sigma^*$, a set of *instances*.

(2) For each instance $\varphi \in I(L)$, we associate it with the following:

(i) $S_L(\varphi)$, a finite subset of Σ^* called the set of *feasible solutions*. An element s in $S_L(\varphi)$ is called a *solution* of φ.

(ii) $N_L(\varphi, s)$, a subset of $S_L(\varphi)$ called the *neighbors* of s, where s is a solution in $S_L(\varphi)$. We call a solution in $N_L(\varphi, s)$ a *neighbor* of s.

(iii) $M_L : I(L) \times S_L(\varphi) \rightarrow \mathrm{N}$, the *measure function* for $S_L(\varphi)$, where N is the set of nonnegative integers. The value $M_L(\varphi, s)$ is called the *measure* of s.

We require that $I(L), S_L, N_L$ and M_L are polynomial-time computable with respect to $|\varphi|$. A solution $s \in S_L(\varphi)$ is called *locally optimal* if s has no better neighbors, i.e., $M_L(\varphi, s') \leq M_L(\varphi, s)$ for all s' in $N_L(\varphi, s)$ when L is a maximization problem. In addition, the following two polynomial-time algorithms must exist for L: (i) $Initial_L$, given $\varphi \in I(L)$, produces any feasible solution in $S_L(\varphi)$, and (ii) $Improve_L$, given $\varphi \in I(L)$ and $s \in \Sigma^*$, produces a better neighbor if s is in $S_L(\varphi)$ and not locally optimal, otherwise simply returns s. The algorithms $Initial_L$ and $Improve_L$ allow us to apply the local search algorithm.

For the class of PLS problems, the PLS-reductions are defined as follows:

Definition 3. Let L and K be problems in PLS. We say that L is PLS-*reducible* to K if there be polynomial-time computable functions f and g such that for each instance φ of L, (i) $f(\varphi)$ is an instance of K, (ii) $g(f(\varphi), s)$ is a solution of φ if s is a solution of $f(\varphi)$, and (iii) if $s \in S_K(f(\varphi))$ is a locally optimal solution of $f(\varphi)$, then $g(f(\varphi), s) \in S_L(\varphi)$ is also a locally optimal solution of φ.

We say a problem K is PLS-*complete* if every PLS-problem is PLS-reducible to K. Notice that the PLS-reducibility is transitive, and if we can find locally optimal solutions for a PLS-complete problem in polynomial time then we can also find locally optimal solutions for any PLS problem in polynomial time. There are some amount of PLS-complete problems which are related to well known NP-complete decision problems. There are some problems whose number of variations of measures are polynomially bounded and thus locally optimal solutions can be found in polynomial-time if the weights given in instances are restricted to constant. There are well known local search problems which are PLS-complete, such as Lin-Kernighan algorithm for TSP [11], k-opt change algorithm for TSP [8], the clause weighted satisfiability problems by 1-bit flipping neighborhood [7], and some are found in [15].

Now we define the polynomial-time local search problem for finding pseudo-indexings. For the sake of concentrating on hardness of the problem, we restrict training examples be the whole sets of examples, i.e., we use all examples as training examples.

Definition 4. The *alphabet indexing problem by local search* (AILS) is a maximization problem given by an instance $\pi = (\Sigma, \Gamma, P, Q, \Omega^P, \Omega^Q, A)$, where Σ and Γ are finite alphabets with $|\Sigma| > |\Gamma|$, P and Q are sets of strings over Σ, Ω^P, Ω^Q are the weight functions from each P and Q to nonnegative integers, and A is a robust algorithm that takes the information gain function as the goodness function.

The set $S(\pi)$ of feasible solutions is the set of all mappings from Σ to Γ. The measure of solutions $M(\pi, \psi)$ for $\psi \in S(\pi)$ for A are defined as follows. Let h be the hypothesis obtained by the algorithm A for the input $(\tilde{\psi}(P), \tilde{\psi}(Q); \Omega^{\psi,P}, \Omega^{\psi,Q})$. We compute

$$M(\pi, \psi) = \xi(\sum_{\tilde{\psi}(p) \in \tilde{P} \cap L_h} \Omega^P(p), \sum_{\tilde{\psi}(p) \in \tilde{P} \cap \overline{L_h}} \Omega^P(p), \sum_{\tilde{\psi}(q) \in \tilde{Q} \cap L_h} \Omega^Q(q), \sum_{\tilde{\psi}(q) \in \tilde{Q} \cap \overline{L_h}} \Omega^Q(q)),$$

where L_h is the language defined by h and $\xi(x, x'y, y') = \max(x \cdot y', x' \cdot y)$. The neighborhood $N(\pi, \psi)$ for $\psi \in S(\pi)$ is $N(\pi, \psi) = \{\phi \in S(\pi) \mid \delta(\phi, \psi) = 1\}$, where $\delta(\phi, \psi) = |\{\sigma \in \Sigma \mid \phi(\sigma) \neq \psi(\sigma)\}|$.

Although the measure function ξ in the above definition is fixed to the one-sided multiplication, the Theorem 2 holds for any polynomial-time computable function satisfying the following conditions: $\xi(a, a', b, b') = \xi(a', a, b', b) = \xi(b', b, a', a)$ for any

$0 \leq a, a', b, b'$ and $\xi(p, p', q, q') < \xi(p+\varepsilon, p'-\varepsilon, q, q')$ for any $0 \leq p' \leq p, 0 \leq \varepsilon \leq q \leq q'$, $\dfrac{q}{q+q'} \leq \dfrac{p}{p+p'}$.

Theorem 2. The alphabet indexing problem by local search is PLS-complete, if $P, Q \subseteq \Sigma^l$ for $l \geq 4$. If the weights for all strings are polynomially bounded to the size of the instance and the length of the strings is unbounded, the problem is P-complete.

Proof. An instance of the polynomial-time local search problem k-SAT is a boolean formula in k-CNF with a nonnegative integer weight on each clause. A solution is a boolean assignment to the variables, and its measure is the sum of the weights of all satisfied clauses. The neighborhoods of a solution are assignments of Hamming distance 1. This neighborhood structure is called FLIP [5]. It is known that k-SAT (WEIGHTED k-CNF SATISFIABILITY) for $k \geq 2$ is PLS-complete and P-complete if the weights of the clauses are polynomially bounded [15]. We show the polynomial-time local search problem 2-SAT [15] is PLS-reducible to the alphabet indexing problem by local search. We say that instances of k-SAT and AILS are *unweighted* if their weights on clauses or strings are polynomially bounded for their sizes, and *weighted* if otherwise. We first show a reduction in the weighted case, then we modify the reduction for the unweighted case.

Weighted case: Given a 2-CNF formula $\phi = (X, C)$ with variables $X = \{x_1, \ldots, x_n\}$ and weighted clauses $C = \{c_1, \ldots, c_m\}$, we define f as follows. We define the alphabets $\Sigma = \{x_i \mid 1 \leq i \leq n\} \cup \{\mathtt{t}, \mathtt{f}\}$ and $\Gamma = \{0, 1\}$. The sets of strings P, Q are defined as follows. Let $c_j = (l_1^j, l_2^j)$ be a clause in ϕ. If l_k^j is a positive literal x_i, let $t_k^j = x_i$ and $f_k^j = \mathtt{t}$. Otherwise, let $t_k^j = \mathtt{f}$ and $f_k^j = x_i$. Then we define $P = \{f_1^j t_1^j f_2^j t_2^j \mid (l_1^j, l_2^j) \in C\} \cup \{\mathtt{tttt}\}$ and $Q = \{\mathtt{tftf}\}$. Finally, we define the weight function Ω^P and Ω^Q. The weight of a string corresponding to c_j in P is the weight of c_j and the weight of $\mathtt{tttt}$ is $\omega + 1$, where ω is the sum of the weights of all clauses in ϕ. The weight of $\mathtt{tftf}$ in Q is $2\omega + 2$. Next we define a mapping g from pseudo-indexings to boolean assignments. For any pseudo-indexing ψ of $f(\phi)$, we define an assignment $g(f(\phi), \psi)$ as follows. If $\psi(\mathtt{t}) = 1$, we give the boolean value represented by $\psi(x_i)$ for each variable x_i with $1 \leq i \leq n$. Otherwise, we give $x_i = \neg\psi(x_i)$. It is clear that this mapping gives a boolean assignment for ϕ.

The remaining part of the reduction is to show that if an indexing ψ is locally

optimal then $g(f(\phi), \psi)$ is locally optimal for ϕ. To do this, we show the following lemma.

Lemma 1. Any locally optimal indexing satisfies $\psi(\mathtt{t}) \neq \psi(\mathtt{f})$.

Proof. Assume that a locally optimal indexing ψ satisfies $\psi(\mathtt{t}) = \psi(\mathtt{f})$. This implies that $\tilde{\psi}(\mathtt{tttt})$ is identical to $\tilde{\psi}(\mathtt{tftf})$. Since Q has only one string, the goodness value of a hypothesis is maximized when the total weight of converted strings $\tilde{\psi}(P) - \tilde{\psi}(Q)$ is maximized. Since the measure function for indexings is symmetric, we can assume that A produces h which classifies $\tilde{\psi}(\mathtt{tttt}), \tilde{\psi}(\mathtt{tftf}) \notin L_h$ without loss of generality. Therefore the measure of ψ is at most $\xi(\omega, \omega+1, 0, 2\omega+2) = \omega \cdot (2\omega+2)$, when h classifies all strings of P except $\tilde{\psi}(\mathtt{tttt})$ in L_h. On the other hand, we can distinguish $\tilde{\psi}(\mathtt{tttt})$ from $\tilde{\psi}(\mathtt{tftf})$ by any indexing ψ' satisfying $\psi'(\mathtt{t}) \neq \psi'(\mathtt{f})$. Such indexings have the measure at least $\xi(\omega+1, \omega, 0, 2\omega+2) = (\omega+1) \cdot (2\omega+2)$. By flipping $\psi(\mathtt{t})$ or $\psi(\mathtt{f})$, we can obtain a solution ψ' satisfying this condition, and the measure of the new solution ψ' overcomes the measure of ψ. This contradicts the assumption. Therefore any locally optimal indexing satisfies $\psi(\mathtt{t}) \neq \psi(\mathtt{f})$. □

Now, we show that if an indexing is locally optimal then the boolean assignment for ϕ obtained by g is also locally optimal. Let ψ be a locally optimal indexing, and let W be the sum of the weights of the strings corresponding to clauses that have two adjoining same symbols $\tilde{\psi}(\mathtt{tt})$ and/or $\tilde{\psi}(\mathtt{ff})$ after the transformation by ψ. The strings corresponding to clauses can have a such adjoining same symbol only if they have a variable symbol x_i satisfying $\tilde{\psi}(\mathtt{t}x_i) = \tilde{\psi}(\mathtt{tt})$ or $\tilde{\psi}(x_i\mathtt{f}) = \tilde{\psi}(\mathtt{ff})$, i.e., the clause has x_i satisfying $\psi(x_i) = \psi(\mathtt{t})$ for a positive literal or $\psi(x_i) = \psi(\mathtt{f})$ for a negative literal. Since ψ is a locally optimal solution, ψ satisfies $\psi(\mathtt{t}) \neq \psi(\mathtt{f})$, and thus any string that has two adjoining same symbols can be distinguished from $\mathtt{tftf}$. Therefore, W is equal to the sum of the weights of clauses satisfied by the assignment $g(f(\phi), \psi)$, and the measure of ψ is $\xi(W+\omega+1, \omega-W, 0, 2\omega+2) = (W+\omega+1) \cdot (2\omega+2)$. Suppose that ψ is locally optimal but the assignment $g(f(\phi), \psi)$ is not. Then the assignment must have a flip for improving the solution that has larger measure, say W'. This implies that ψ has also an improved neighbor, since the same flip for the same variable symbol of ψ improves the measure to $\xi(W'+\omega+1, \omega-W', 0, 2\omega+2) = (W'+\omega+1) \cdot (2\omega+2)$. This contradicts the assumption. Thus the theorem holds in weighted case.

Unweighted case: Let $\phi = (X, C)$ be a 2-CNF formula with variables $X = \{x_1, \ldots, x_n\}$ and clauses $C = \{c_1, \ldots c_m\}$ with constant weight one. We transform the unweighted formula to an unweighted AILS as follows.

Let Σ, Γ be the alphabets constructed as in the weighted case. For each clause c_i in ϕ, we make $f_1^i t_1^i f_2^i t_2^i$ and add it to P. We add all $\{\mathtt{tttt}(\mathtt{tf})^i \mid 0 \leq i \leq m\}$ to P, and add all $\{\mathtt{tftf}(\mathtt{tf})^i \mid 0 \leq i \leq m\}$ to Q. The mapping g is defined as in the weighted case. Then any locally optimal indexing of this instance also satisfies Lemma 1. If an indexing ψ satisfies $\psi(\mathtt{t}) = \psi(\mathtt{f})$, the measure of ψ is at most $\xi(m, m+1, 0, m+1)$ for an output of A classifying $\tilde{\psi}(\mathtt{tttt}), \tilde{\psi}(\mathtt{tftf}) \notin L_h$ and the all strings corresponding to clauses in L_h. On the other hand, if $\psi(\mathtt{t}) \neq \psi(\mathtt{f})$, the measure is at least $\xi(m+1, m, 0, m+1)$. If an indexing is locally optimal, any string corresponding to a clause is distinguished from $\mathtt{tftf}$ after the transformation if and only if it has two contiguous same symbols. Therefore, as in the weighted case, the boolean assignment given by g is locally optimal if the indexing is locally optimal. □

4 Conclusions

We have defined an alphabet indexing which is strictly consistent with positive and negative examples. In such case, we have shown that finding an indexing is NP-complete. We have also defined the concept of a pseudo-indexing, and shown that the problem of finding a locally optimal pseudo-indexing by the algorithm in reference [16] is PLS-complete. Even when the strings are unweighted, the result states that finding a locally optimal pseudo-indexing is P-complete.

As we have shown in this paper, finding a good indexing is computationally hard. However, by applying suitable indexing to sequences of amino acid residues, we could reduce the computation time and space for finding hypotheses. Moreover, a good indexing may reduce the hypothesis spaces and simplify the hypotheses. The experimental results in reference [16] support these assumptions. For taking these advantages and dealing with the hardness of the problem, we can regard some polynomial-time approximation algorithm. For example, we have P-complete algorithms for (1) finding smaller size of indexing alphabets and an indexing which retains $\tilde{\psi}(P) \cap \tilde{\psi}(Q) = \emptyset$, and (2) finding a pseudo-indexing which minimizes $|\tilde{\psi}(P) \cap \tilde{\psi}(Q)|$. The recent researches on polynomial-time approximation algorithms [12] assert that

we have polynomial-time approximation algorithms for subproblems of AIP, which can obtain constantly error-bounded solutions. Estimating such error ratio for those polynomial algorithms, and establishing precise boundaries of the problems for P and NP completeness are left open as the future work.

Acknowledgments

The authors would like to thank S. Arikawa, T. Shinohara, S. Kuhara and A. Shinohara for their many helpful discussions with this work. Also, a special thanks to R. Greenlaw and the referees for the encouragement and comments for the results.

This work is partly supported by Grant-in-Aid for Scientific Research on Priority Areas, "Genome Informatics" from the Ministry of Education, Science and Culture, Japan.

References

[1] Arikawa, S., Kuhara, S., Miyano, S., Shinohara, A. and Shinohara, T., "A learning algorithm for elementary formal systems and its experiments on identification of transmembrane domains," in *Proc. 25th Hawaii International Conference on System Sciences*, pp. 675–684, 1992

[2] Arikawa, S., Miyano, S., Shinohara, A., Kuhara, S., Mukouchi, Y. and Shinohara, T., "A machine discovery from amino acid sequences by decision trees over regular patterns," *New Gener. Comput.*, vol. 11, pp. 361–375, 1993

[3] Arikawa, S., Shinohara, T. and Yamamoto, A., "Learning elementary formal systems," *Theor. Comput. Sci.*, vol. 95, pp. 97–113, 1992

[4] Garey, M. R. and Johnson, D. S., *Computers and Intractability: A Guide to the Theory of NP Completeness*, W. H. Freeman and Company, New York, 1978

[5] Johnson, D. S., Papadimitriou, C. H. and Yannakakis, M., "How easy is local search?," *J. Comput. Sys. Sci.*, vol. 37, pp. 79–100, 1988

[6] Jones, N. and Lasser, T., "Complete problems for deterministic polynomial time," *Theor. Comput. Sci.*, vol. 3, pp. 105–117, 1977

[7] Krentel, M. W., "On finding locally optimal solutions," *SIAM J. Comput.*, vol. 19, pp. 742–749, 1990

[8] Krentel, M. W., "On Finding Locally Optimal Solutions (extended abstract)," in *Proc. IEEE Structure in Complexity Theory Fourth Annual Conference*, pp. 132–137, 1989

[9] Kyte, J. and Doolittle, R. F., "A simple method for displaying the hydropathic character of protein," *J. Mol. Biol.*, vol. 157, pp. 105–132, 1982

[10] Lander, R. E., "The circuit value problem is log space complete for P," *SIGACT News*, vol. 7, pp. 18–20, 1975

[11] Papadimitriou, C. H., "The complexity of the Lin-Kernighan heuristic for the traveling salesman problem," *SIAM J. Comput.*, vol. 21, no. 3, pp. 450–465, 1992

[12] Papadimitriou, C. H. and Yannakakis, M., "Optimization, approximation, and complexity classes" *J. Comput. Sys. Sci.*, vol. 43, pp. 425–440, 1991

[13] PIR, protein identification resource, National Biomedical Research Foundation

[14] Quinlan, J., "Induction of decision trees," *Machine Learning*, vol. 1, pp. 81–106, 1986.

[15] Schäffer, A. A. and Yannakakis, M., "Simple local search problems that are hard to solve," *SIAM J. Comput.*, vol. 20, no. 1, pp. 56–87, 1991

[16] Shimozono, S., Shinohara, A., Shinohara, T., Miyano, S., Kuhara, S. and Arikawa, S., "Finding alphabet indexing for decision trees over regular patterns," in *Proc. 26th Annual Hawaii International Conference on System Sciences*, pp. 763–772, 1993

[17] Watson, J. D., Hopkins, N. H., Roberts, J. W., Steitz, J. A. and Weiner, A. M., *Molecular biology of the gene*, fourth edition, The Benjamin/Cummings Publishing Company, Menlo Park, California, 1987

[18] Yanagihara, N., Suwa, M. and Mitaku, S., "A theoretical method for distinguishing between soluble and membrane proteins," *Biophysical Chemistry*, vol. 34, no. 1, pp. 69–77, 1989