Inferring a Tree from Walks

Maruyama, Osamu
Department of Information Systems, Kyushu University

Miyano, Satoru
Research Institute of Fundamental Information Science Kyushu University

出版情報：RIFIS Technical Report. 51, 1991-12-05. Research Institute of Fundamental Information Science, Kyushu University
バージョン：
権利関係：

RIFIS-TR-CS-51

RIFIS Technical Report

Inferring a Tree from Walks

Osamu Maruyama
Satoru Miyano

December 5, 1991
Revised: March 24, 1995

Research Institute of Fundamental Information Science
Kyushu University 33
Fukuoka 812, Japan
E-mail: maruyama@rifis.kyushu-u.ac.jp Phone: +81-92-641-1101 ex.4478

Inferring a Tree from Walks[§]

Osamu Maruyama[†,*] and Satoru Miyano[‡]

[†]Department of Information Systems, Kyushu University 39, Kasuga 816, Japan

[‡]Research Institute of Fundamental Information Science

Kyushu University 33, Fukuoka 812, Japan

Abstract

A walk in an undirected edge-colored graph G is a path containing all edges of G. The tree inference from a walk is, given a string x of colors, finding the smallest tree that realizes a walk whose sequence of edge-colors coincides with x. We prove that the problem is solvable in $O(n)$ time, where n is the length of a given string. We furthermore consider the problem of inferring a tree from a finite number of partial walks, where a partial walk in G is a path in G. We show that the problem turns to be NP-complete even if the number of colors is restricted to 3. It is also shown that the problem of inferring a linear chain from partial walks is NP-complete, while the linear chain inference from a single walk is known to be solvable in polynomial time.

1. Introduction

A walk in an undirected edge-colored graph G is a path that contains all edges of G. For a walk w, the trace of w is the string of edge-colors seen in w. Aslam and Rivest [3] asked: Given a string x of colors and a positive integer k, what is an undirected, degree-bound k, edge-colored graph G with the minimum number of edges such that G realizes a walk with trace x? Rudich [14] has discussed a problem closely related to the graph inference. He considered the problem of inferring a Markov chain from its output, and developed algorithms that for the binary output of a Markov chain, in the limit, reconstruct the underlying Markov chain structure as well as the associated transition probabilities. Aslam and Rivest [3] settled the problem of inferring graphs of bounded degree 2 (linear chains and cycles) from a walk, by proving that a certain set of rewriting rules satisfies the Church-Rosser or confluence property. They established $O(n^3)$ and $O(n^5)$ time

[§]A preliminary version of this work was presented at the *Seventeenth Symposium on Mathematical Foundations of Computer Science* (MFCS '92), Prague, Czechoslovakia, 1992.

*Corresponding author. Research Institute of Fundamental Information Science, Kyushu University 33, Fukuoka 812, Japan. Email: maruyama@rifis.kyushu-u.ac.jp. This author is a Research Fellow of the Japan Society for the Promotion of Science (JSPS).

algorithms for finding the smallest linear chain and cycle consistent with a given string of colors, respectively, where n is the length of the string. The latter bound has been improved by Raghavan [13] to $O(n \log n)$ time. However, he additionally showed that for all $k \geq 3$, the problem of inferring a graph of bounded degree k with the minimum number of nodes is NP-complete.

This paper solves the problem for trees of unbounded degree. The *tree inference from a walk* is the problem of finding the smallest undirected edge-colored tree that has a trace coinciding with a given string of colors. We give an $O(n)$ time algorithm for the problem. Recently, Maruyama and Miyano [9] have shown that the problem of inferring a tree of bounded degree k from a walk is NP-complete for $k \geq 3$ even if the number of colors is $k+1$.

A partial walk in an undirected edge-colored graph G is a path in G, while a walk in G must contain all edges of G. We then ask: Given a finite set S of strings, what is an undirected edge-colored tree T with the minimum number of edges such that, for each $x \in S$, T has some partial walk with trace x. We call this problem the *tree inference from partial walks*. In contrast with the case of a single walk, we prove that the tree inference from partial walks turns to be NP-complete even if all strings of S are written over an alphabet of size 3.

We next consider the problem of inferring a linear chain from partial walks. Similarly, we show that this problem is also NP-complete even if the size of alphabet is 3, while the linear chain inference from a single walk is solvable in polynomial time [3, 13].

Given a finite set of strings over an alphabet of size at most 2, we show that the tree inference from partial walks and the linear chain inference from partial walks are solvable by the same algorithm in linear time. In order to show the NP-hardness of the linear chain inference from partial walks, we give a reduction from the shortest common superstring problem [5]. It is interesting that although the shortest common superstring problem is NP-complete even if the size of alphabet is restricted to 2, yet the linear chain inference from partial walks is solvable in linear time if the size of alphabet is 2.

The problem of identifying the smallest finite automaton consistent with given input/output behaviors, which is shown to be, in general, NP-complete by Angluin [1] and Gold [7], is a problem similar to these graph inference problem. The identification problem can be regarded as the case that a *directed* edge-colored graph is to be inferred from strings. Moreover, Pitt and Warmuth [12] have shown an interesting negative result on approximation algorithms for the problem. We show that there is an approximation algorithm for the tree inference from partial walks which is constructed by employing an algorithm that approximately solves the minimum common supertree problem [15]. We next give polynomial-time approximation algorithms for the linear chain inference from partial walks which employ algorithms that approximate the problem of shortest superstrings with flipping [8]. We furthermore show that these inference problems are MAXSNP-hard, which implies that there are no polynomial-time approximation schemes for the problems unless P=NP [2].

This paper is organized as follows. In Section 2, we introduce some basic definitions to be used throughout the paper. In Section 3, it is proved that the tree inference from a walk is solvable in $O(n)$ time. In Section 4, we show that the tree inference from partial walks is NP-complete and the linear chain inference from partial walks is also NP-complete. Finally, we give results on approximabilities of these intractable problems in Section 5.

2. Preliminaries

Let Σ be a finite alphabet. The set of all strings over Σ is denoted by Σ^*. For a string x, the length of x is denoted by $|x|$ and the reversal of x is written as x^R. The concatenation of strings x and y is written as $x \cdot y$, or simply xy. For strings $x_1, \ldots, x_n$, $\prod_{i=1}^{n} x_i$ denotes $x_1 x_2 \cdots x_n$. If S is a set, $|S|$ denotes the cardinality of S.

A *color* is a symbol in Σ. In this paper we consider undirected edge-colored graphs $G = (V, E, c)$, where $c : E \to \Sigma$ is called the *edge-coloring* of G. Hereafter a graph means an undirected edge-colored graph without any notice. For graphs G and G', if G and G' are isomorphic including edge labels, we identify G with G' without any notice. A graph G is said to be *proper* if no two adjacent edges have the same color. A *linear chain* is a graph $l = (V, E, c)$ with $V = \{v_i \mid i = 1, \ldots, m\}$ and $E = \{\{v_i, v_{i+1}\} \mid i = 1, \ldots, m-1\}$, and the *label* of l is defined as the string $\prod_{i=1}^{m-1} c(\{v_i, v_{i+1}\})$. Note that for any string x, a linear chain with label x is identified with a linear chain with label x^R. We denote the classes of linear chains, trees and graphs of bounded degree k by LinearChain, Tree and k-Deg, respectively.

A *partial walk* in a graph G is a path in G. If a partial walk in G contains all edges of G, it is called a *walk* in G. For the sequence $e_1, e_2, \ldots, e_n$ of edges in a partial walk w in $G = (V, E, c)$, the *trace* of w is defined as the string $\prod_{i=1}^{n} c(e_i)$. Let x be a string. If w is a (partial) walk with trace x, w is called a *(partial) walk for* x. For a graph G, we say that G *realizes* a walk for x if there is a walk for x in G. Similarly, for a graph G and a finite set S of strings, we say that G *realizes all partial walks for* S if for each $x \in S$, there is a partial walk for x in G.

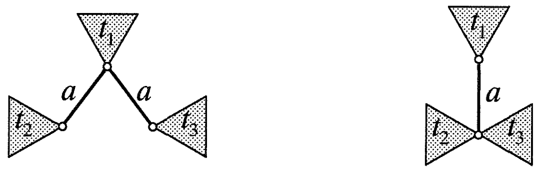
Let $\to$ be a binary relation on a set D and $\xrightarrow{*}$ be the transitive and reflexive closure of $\to$. For $x, y \in D$, if $x \xrightarrow{*} y$ and there is no $z \in D$ such that $y \to z$ then y is called a $\to$-*normal form* of x.

Definition. Let $T_1 = (V, E, c)$ be a tree which includes adjacent edges $e_1 = \{v_1, v_2\}$ and $e_2 = \{v_2, v_3\}$ with $c(e_1) = c(e_2)$ (see Fig. 1 (a)). Let T_2 be the tree obtained from T_1 by identifying v_3 with v_1 together with the adjacent edges e_1 and e_2 (see Fig. 1 (b)). Then we say that T_2 is an *edge-folding of* T_1. The binary relation $\to_F$ on the set of trees is defined to be the set of pairs (T_1, T_2) such that T_2 is an edge-folding of T_1.

Fact 1. For trees T_1 and T_2, suppose that $T_1 \to_F T_2$. The following facts hold trivially:

1. T_2 is smaller than T_1.

2. If T_1 realizes a walk for a string x, then T_2 realizes a walk for x.



(a) T_1 (b) T_2

Figure 1: t_1, t_2 and t_3 in (a) and (b) are arbitrary trees and a is an arbitrary color.

3. If T_1 realizes all partial walks for a set S of strings, then T_2 realizes all partial walks for S.
4. For a tree T, an $\rightarrow_F$-normal form of T is proper.

3. Inferring a tree from a walk

In this section, we give a linear-time algorithm for finding the smallest tree realizing a walk for a given string. The *tree inference from a walk* is defined as follows:

Instance: A string x over a finite alphabet Σ.

Problem: Find a tree T with the minimum number of edges such that T realizes a walk for x.

Theorem 1. The tree inference from a walk is solvable in $O(n)$ time, where n is the length of a given string.

Assume that a tree T realizes a walk for a string x. If T is not proper, then there is an edge-folding T' of T. We can see by Fact 1 that T' is smaller than T and realizes a walk for x. Thus we can have the following lemma:

Lemma 1. For a string x, any of the smallest trees realizing a walk for x is proper.

Given a string x, one way to make a proper tree that realizes a walk for x is repeating the following procedure: Let v_i be the end node of a walk for the prefix of x with length i realized in the resulting proper tree just after the ith iteration. If v_i does not have any adjacent edge labeled x_{i+1}, where x_{i+1} is the $i+1$st symbol of x, then, using a new node u, the edge $\{u, v_i\}$ labeled x_{i+1} is created and let $v_{i+1} := u$. Otherwise, let $v_{i+1} := u$, where $\{u, v_i\}$ is an edge labeled x_{i+1}. Obviously, the tree produced in this procedure is a proper tree realizing a walk for x. Moreover, we can easily check the following lemma:

Lemma 2. For any string x, a proper tree realizing a walk for x is unique.

Note that this result implies that for a string x, an $\rightarrow_F$-normal form of a linear chain with label x is unique. The following algorithm, called EDGE-FOLD, is based on the above idea. The tree produced by the algorithm is represented by an array

T indexed on the vertices and the colors. We can consider that the vertices and the colors are coded into the numbers.

```
/* x = x_1 ··· x_n (x_i ∈ Σ) */
begin
    u := 1; v := 1;
    for i := 1 to n
        if T[u, x_i] = 0  then
            v := v + 1;
            T[u, x_i] = v; T[v, x_i] := u; /* {u, v} is an edge labeled x_i */
            u := v;
            else u := T[u, x_i]
        endif
    end;
    return T
end;
```

Algorithm : EDGE-FOLD

It is clear that the algorithm EDGE-FOLD always produces a proper tree realizing a walk for a given string. Thus, by Lemmas 1 and 2, the tree produced by EDGE-FOLD is the smallest tree that realizes a walk for a given string. The number of steps executed by every iteration of the loop of EDGE-FOLD is bounded by a constant. Thus EDGE-FOLD runs in $O(n)$ time.

4. Inferring a graph from partial walks

Instead of dealing with a single walk, we consider in this section, the problem of inferring a tree from a finite number of partial walks. We consider the following decision problem:

Definition. Let C be a class of graphs. The *graph inference from partial walks for C*, denoted by GIPWS(C), is defined as follows:

Instance: A finite set S of strings over a finite alphabet Σ and a positive integer K.

Question: Is there a graph G in C with at most K edges such that G realizes all partial walks for S?

The *tree inference from partial walks* is defined as GIPWS(Tree). The main result in this section is the following theorem:

Theorem 2. The tree inference from partial walks is NP-complete. Furthermore, this problem is NP-complete even if the size of alphabet is restricted to 3.

Proof. It is easy to see that GIPWS(Tree) is in NP. We first reduce the vertex cover problem [6] to GIPWS(Tree), where the vertex cover problem (VC) is to decide if, given a graph $G = (V, E)$ and a positive integer K, there is a vertex cover of size at most K for G, that is, a subset $C \subseteq V$ with $|C| \leq K$ such that for each edge $\{u, v\} \in E$ at least one of u and v belongs to C. After that, we modify the reduction so as to show that the problem remains NP-complete if the size of alphabet is restricted to 3.

Let $G = (V, E)$ be a graph with $|V| = n$ and K be a positive integer. For G and K, We define an alphabet Σ as $\Sigma = V \cup \{a_0, a_1, \ldots, a_{\lceil n/2 \rceil}\} \cup \{b_1, b_2, \ldots, b_{n+1}\}$. In order to define a set S of strings over Σ, we introduce the following notations for strings:

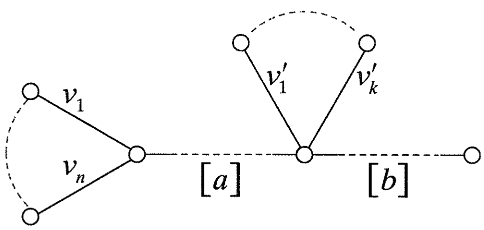
$$\begin{aligned} [a] &= a_{\lceil n/2 \rceil} \cdots a_1 a_0 a_1 \cdots a_{\lceil n/2 \rceil} \\ [b] &= b_1 \cdots b_{n+1} \, . \end{aligned}$$

Note that $[a]^R = [a]$. Then S consists of the following strings:

$$\begin{array}{ll} \textit{base-string}: u[a][b] & \text{for } u \in V, \\ \textit{edge-string}: u[a]v & \text{for } \{u, v\} \in E. \end{array}$$

Finally, let $K' = 2n + 2\lceil n/2 \rceil + 2 + K$. This transformation can be done in polynomial time. We claim that G has a vertex cover of size at most K if and only if there is a tree with at most K' edges which realizes all partial walks for S.

Suppose that G has a vertex cover C with $|C| \leq K$. For a subset $U = \{v'_1, \ldots, v'_k\}$ of V, let $T(U)$ be the tree in Fig. 2. It is obvious that $T(C)$ re-



$T(U)$

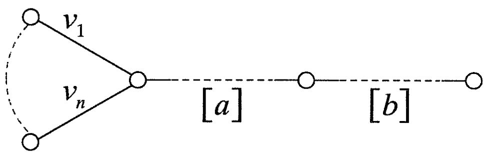
Figure 2: $V = \{v_1, \ldots, v_n\}$ and $U = \{v'_1, \ldots, v'_k\} \subseteq V$.

alizes all partial walks for S. It can be easily checked that T contains at most K' edges since $|C| \leq K$.

Conversely, suppose that there is a tree T with at most K' edges realizing all partial walks for S. Note that for $x \in S$, any tree realizing a walk for x is isomorphic to a linear chain with label x. Without loss of generality, we can assume that T is proper by Fact 1. Note that if T is proper then any subgraph of T is proper.

We first consider the base-strings, each of which includes exactly one $[a][b]$ as a substring.

Claim 1. Let T_b be the tree in Fig. 3. Any proper tree with at most K' edges that realizes all partial walks for the set of the base-strings is isomorphic to the tree T_b.



T_b

Figure 3: $V = \{v_1, \ldots, v_n\}$.

Proof. It can be easily checked that if such a tree is not isomorphic to T_b then it contains at least $|[a][b]| + |[b]| + |V| = 3n + 2\lceil n/2 \rceil + 3$ edges. This contradicts the assumption that the number of edges in T is at most K'. $\square$

In a similar way, we can see the following:

Claim 2. For a tree T', T' is a proper tree with at most K' edges realizing all partial walks for S if and only if T' is isomorphic to the tree $T(C')$ where $C' \subseteq V$.

Then we can assume that for some $C' \subseteq V$, the tree T is isomorphic to $T(C')$. It is obvious that $|C'|$ is at most K since T contains at most K' edges. It should be clear that C' gives a vertex cover of G whose size has been shown at most K.

We next modify the reduction into another one to show that the tree inference from a walk remains NP-complete if the size of alphabet is restricted to 3. Let $\Sigma = \{0, 1, \#\}$. For convenience, we asssume that $V = \{0, \ldots, n-1\}$. For a nonnegative integer i, we denote by i_j the jth bit of the binary representation of i such that $i = i_0 2^0 + i_1 2^1 + \cdots + i_{m-1} 2^{m-1}$ for some $m \geq \lfloor \log i \rfloor + 1$. Let $\overline{i_j} = 1$ if $i_j = 0$ and $\overline{i_j} = 0$ otherwise. For a pair (h, i) of integers with $0 \leq i \leq 2^h - 1$, the strings $b_1(h,i), b_2(h,i)$ and $b_3(h,i)$ are defined as follows:

$$\begin{aligned}
b_1(h,i) &= \#i_0\#i_1 \cdots \#i_{h-1}. \\
b_2(h,i) &= \#i_0\overline{i_0}\#i_1\overline{i_1} \cdots \#i_{h-1}\overline{i_{h-1}}. \\
b_3(h,i) &= \#i_0\overline{i_0}i_0\#i_1\overline{i_1}i_1 \cdots \#i_{h-1}\overline{i_{h-1}}i_{h-1}.
\end{aligned}$$

Let $q = \lceil \log n \rceil$. Using these strings, we make the strings $[i]$ for $0 \leq i \leq 2^q - 1$, $[a]$ and $[b]$ as follows:

$$\begin{aligned}
[i] &= b_1(q,i) \text{ for } 0 \leq i \leq 2^q - 1. \\
\tilde{a} &= \prod_{i=0}^{2^q-1} \#0101 b_2(q,i). \\
[a] &= \tilde{a}\#0\#\tilde{a}^R. \\
[b] &= \prod_{i=0}^{2^q-1} 01010101 b_3(2q,i)\#.
\end{aligned}$$

Note that $||[a]|| = 2^{q+1}(3q+5)+3$ and $||[b]|| = 2^{q}(8q+9)$. The strings of S are defined as follows:

base-string :	$[i][i][a][b]$	for $i \in V$.
branch-string :	$[i][a][i]^R$	for $0 \leq i \leq 2^q - 1$.
edge-string :	$[i][i][a][j]^R[j]^R$	for $\{i,j\} \in E$.

Finally, let $K' = 2q(n+K) + 2^q(14q+27) - 6$. This transformation can be done in polynomial time. We claim that G has a vertex cover of size at most K if and only if there is a tree with at most K' edges which realizes all partial walks for S (see Fig.4). This claim can be proven in a similar way of the case that any restriction is not put on the size of alphabet. We leave it for the reader to verify the claim.

□

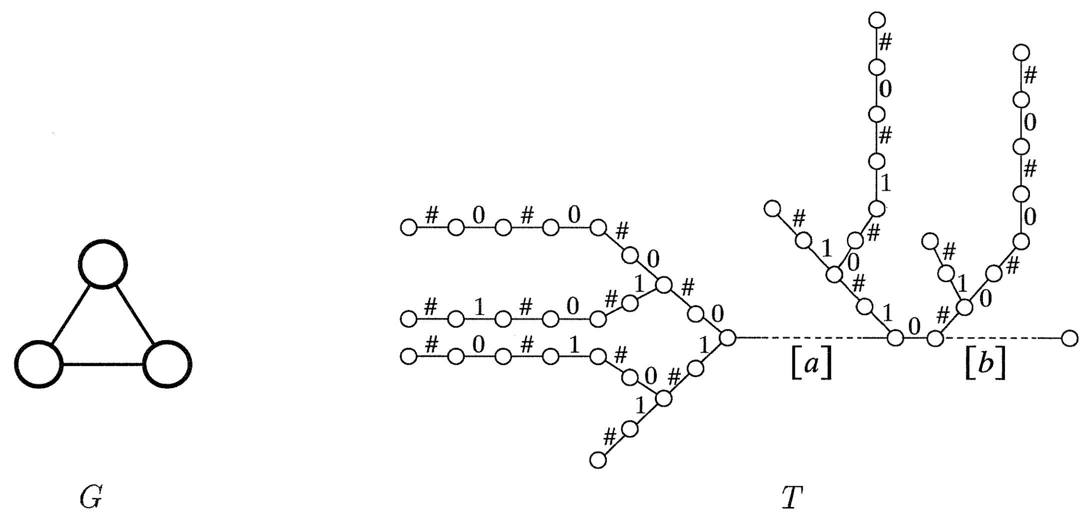


Figure 4: For the graph G, the tree T would be constructed

The *linear chain inference from partial walks* is defined as GIPWS(LinearChain).

Theorem 3. The linear chain inference from partial walks is NP-complete even if the size of alphabet is restricted to 3.

Proof. We give a reduction from the shortest common superstring problem [5], where the shortest common superstring problem is to decide if, given a finite set S of strings over a finite alphabet Σ and a positive integer K, there is a superstring for S with length at most K, that is, a string $s \in \Sigma^*$ with $|s| \leq K$ such that each string $x \in S$ is a substring of s. It is known that the problem is NP-complete even if $|\Sigma| = 2$ [5]. Let S be a finite set of strings over the alphabet $\Sigma = \{0,1\}$ and K be a positive integer. We first define an alphabet Σ' as $\Sigma' = \Sigma \cup \{\#\}$, where # is a new symbol not in Σ. For a string $b = b_1 b_2 \cdots b_m$ with $b_1, b_2, \ldots, b_m \in \Sigma$, we create a string

$$b' = \prod_{i=1}^{m}(01\#b_i\#)01.$$

Then let S' be the set of the strings b' for all $b \in S$. Finally, let $K' = 5K + 2$. This transformation can be done in polynomial time.

Note that the linear chain realizing a walk for a string $x' \in S'$ is the only linear chain $l_{x'}$ with label x' (see Theorem 5 of [3]). It is clear that there is a superstring s for S with $|s| \leq K$ if and only if all partial walks for S' are realized in a linear chain with K' edges or less. □

Theorem 4. The tree inference from partial walks is solvable in linear time if the size of alphabet is at most 2.

Proof. Let Σ be an alphabet of size at most 2. A string x over Σ is said to be alternate if the ith bit of x, denoted by x_i, is different from x_{i+1}. By Lemma 1, the smallest tree realizing a walk for x is a linear chain and the label of it is alternate. We denote the alternate string for x by $a(x)$. For a finite set S of strings over Σ, let $a(S)$ be the set of $a(x)$'s for all $x \in S$. It is obvious that $a(S)$ can be obtained by algorithm EDGE-FOLD in linear time.

Let l be the label of the smallest linear chain that realizes all partial walks for S. The string l is the longest string in $a(S)$ with the exception that the two distinct alternate strings with length $2k+1$ for some k are the longest strings, in which l is the alternate string with length $2k+2$. □

From the proof of Theorem 4, it can be seen that the linear chain inference from partial walks is solvable in linear time if the size of alphabet is at most 2.

5. Approximabilities

As we have shown in Sections 3 and 4 that the graph inferences from partial walks for trees and linear chains are computationally hard, while the inferences from a walk allow polynomial time algorithms. In this section, we discuss the approximabilities of these intractable problems.

First, we yield polynomial-time approximation algorithms for the graph inferences from partial walks for trees and linear chains. An approximation algorithm for the tree inference from partial walks is easily constructed by employing an approximation algorithm for the smallest supertree problem [15]. The approximation ratio of the algorithm for the tree inference depends on that for the smallest supertree problem. Approximation algorithms for the linear chain inference are also constructed by employing approximation algorithms for the shortest common superstring with flipping [8]. Their approximation ratios depend on the ratios of the employed algorithms.

Second, by slightly modifying the reduction in the proof of Theorem 2, the tree inference from partial walks is shown to be MAXSNP-hard, which implies that there is no polynomial-time approximation scheme for the tree inference unless P = NP by the result due to Arora et al. [2]. We can also see that the linear chain inference from partial walks is MAXSNP-hard as a trivial consequence of the reduction in the proof of Theorem 3.

The tree inference from partial walks has the following approximation algorithm that is analyzed in terms of the *compression* in the tree constructed, that is, in terms of $k - l$, where k is the total length of given strings and l is the number of edges of the tree.

Theorem 5. There is a polynomial-time approximation algorithm to find a tree T realizing all partial walks for a set S of strings such that $C \geq C_m/(|S| - 1)$, where C is the compression in T and C_m is the maximum compression for S.

In approximately solving the tree inference from partial walks, the observation in the following lemma is a key to our approach.

Lemma 3. Let T be the smallest tree realizing all partial walks for a set S of strings. Then for each $x \in S$, the smallest tree realizing a walk for x is a subgraph of T.

This lemma is trivial from Lemma 1. For a finite set R of edge-colored trees, an edge-colored tree T is called a *supertree for* R if for $t \in R$, the tree t is a subgraph of T. For a string x, we denote the smallest tree realizing a walk for x by $st(x)$. For a finite set S of strings, $st(S)$ is the set of $st(x)$'s for all $x \in S$. Given a finite set S of strings, if we could find the smallest supertree for $st(S)$, it would be the required tree in the tree inference from partial walks by Lemma 3. Though the problem of finding the smallest supertree is easily seen to be NP-complete from the proof of Theorem 2, there is an approximation algorithm which, given a finite set R of trees, constructs a supertree T for R satisfying $C \geq C_m/(|R| - 1)$, where C is the compression in T and C_m is the maximum compression for R [15]. Thus, by employing the algorithm, the algorithm in Theorem 5 can be given. Notice that for each $x \in S$ if the smallest tree realizing a walk for x is isomorphic to a linear chain with label x, we cannot expect any merit by constructing $st(S)$ in the algorithm of Theorem 5.

We can similarly discuss an approximation algorithm of the linear chain inference from partial walks because we have the following lemma:

Lemma 4. Let l be the smallest linear chain realizing all partial walks for a set S of strings. Then for each $x \in S$, the smallest linear chain realizing a walk for x is a subgraph of l.

This can be easily shown by using binary relations introduced in [3]. A string s is called a *superstring for a set S of strings with flipping* if for each string $x \in S$, either x or x^R is a substring of s. In a similar way, the compression in a superstring with flipping can be defined. Since there is an approximation algorithm which, given a finite set S of strings, find a superstring s with flipping for S such that $C \geq C_m/2$ where C is the compression in s and C_m is the maximum compression for S [8], the following approximation algorithm for the linear chain inference from partial walks is given:

Theorem 6. There is a polynomial-time approximation algorithm to find a linear chain l realizing all partial walks for a set S of strings such that $C \geq C_m/2$, where C is the compression in l and C_m is the maximum compression for S.

Jiang et. al. [8] also developed an approximation algorithm that constructs a superstring s with flipping with length at most $3 \cdot opt$, where opt is the maximum length.

Theorem 7. There is a polynomial-time algorithm to find a linear chain with at most $3 \cdot opt(S)$ edges which realizes all partial walks for a finite set S of strings, where $opt(S)$ is the number of edges in the smallest linear chain realizing all partial walks for S.

We next show that the tree and linear chain inferences from partial walks are MAXSNP-hard. Let Π_1 and Π_2 be two optimization (maximization or minimization) problems. We say that Π_1 *L-reduces* to Π_2 if there are polynomial time algorithms f and g and constants α and $\beta > 0$ such that:

1. Given an instance I_1 of Π_1 with optimal cost $opt(I_1)$, the algorithm f produces an instance I_2 of Π_2 with optimal cost $opt(I_2)$ that satisfies $opt(I_2) \leq \alpha \cdot opt(I_1)$, and

2. Given any feasible solution s_2 of I_2 with cost $cost(s_2)$, the algorithm g produces a solution s_1 of I_1 with cost $cost(s_1)$ such that $|cost(s_1) - opt(I_1)| \leq \beta \cdot |cost(s_2) - opt(I_2)|$.

Some basic facts about L-reductions are: First, the composition of two L-reductions is also an L-reduction. Second, if problem Π_1 L-reduces to problem Π_2 and Π_2 can be approximated in polynomial time with relative error δ, then Π_1 can be approximated with relative error $\alpha\beta\delta$. In particular, if Π_2 has a polynomial-time approximation scheme, then so does Π_1. The class MAXSNP_0 is the class of maximization problems defined syntactically in Papadimitriou and Yannakakis [10, 11]. It is known that every problem in this class can be approximated within *some* constant factor. MAXSNP is defined as the class of all optimization problems that are L-reducible to a problem in MAXSNP_0. A problem is MAXSNP-hard if every problem in MAXSNP can be L-reduced to it.

Theorem 8. The tree inference from partial walks is MAXSNP-hard.

Proof. For an integer k, let k-DEGREE VERTEX COVER be the VC restricted to graphs of bounded degree k. It is known that 4-DEGREE VERTEX COVER is MAXSNP-complete [10, 11]. We can take the reduction in the proof of Theorem 2 as the algorithm f of an L-reduction from 4-DEGREE VERTEX COVER. Then the first condition is satisfied with $\alpha = 15$ since $\lceil n/5 \rceil \leq opt(G)$, where $opt(G)$ is the size of minimum covers of G.

We next define the algorithm g as follows: We can assume that a feasible solution of GIPWS(Tree) is, given a finite set S of strings, a proper tree which realizes all partial walks for S. Let s_2 be a feasible solution of GIPWS(Tree). If s_2 has at most $3n + 2\lceil n/2 \rceil + 2$ edges, then s_2 is a tree isomorphic to $T(C)$ for some $C \subseteq V$, which is defined in the proof of Theorem 2. In the case, the algorithm g returns C. Otherwise, g returns V. Then it is trivial that the second condition holds with $\beta = 1$. □

By the fact that the shortest common superstring problem is MAXSNP-hard [4] and the fact that the reduction in the proof of Theorem 3 is an L-reduction, the following holds:

Theorem 9. The linear chain inference from partial walks is MAXSNP-hard.

Acknowledgments

We would like to thank Ayumi Shinohara for a great amount of helps and suggestions in attacking problems discussed in this paper. We are also grateful to all referees for useful comments. 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. The first author's research is partly supported by Grants-in-Aid for JSPS research fellows from the Ministry of Education, Science and Culture, Japan.

References

[1] D. Angluin. On the complexity of minimum inference of regular sets. *Inform. Control*, 39:337–350, 1978.

[2] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and hardness of approximation problems. In *Proc. 33rd IEEE Symp. Foundations of Computer Science*, pages 14–23, 1992.

[3] J. A. Aslam and R. L. Rivest. Inferring graphs from walks. In *Proc. 3rd Workshop on Computational Learning Theory*, pages 359–370, 1990.

[4] A. Blum, T. Jiang, M. Li, J. Tromp, and M. Yannakakis. Linear approximation of shortest superstrings. *J. Comput. System Sci.*, 41:630–647, 1994.

[5] J. Gallant, D. Maier, and J. A. Storer. On finding minimal length superstrings. *J. Comput. System Sci.*, 20:50–58, 1980.

[6] M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W.H. Freeman and Company, 1979.

[7] E. M. Gold. Complexity of automaton identification from given data. *Inform. Control*, 37:302–320, 1978.

[8] T. Jiang, M. Li, and D. Du. A note on shortest superstrings with flipping. *Inform. Process. Lett.*, 44:195–199, 1992.

[9] O. Maruyama and S. Miyano. Graph inference from a walk for trees of bounded degree 3 is np-complete. Technical report, Research Institute of Fundamental Information Science, Kyushu University, RIFIS-TR-CS **94**, 1994.

[10] C. Papadimitriou and M. Yannakakis. Optimization, approximation and complexity classes. *J. Comput. System Sci.*, 43(3):425–440, 1991.

[11] C. H. Papadimitriou. *Computational Complexity.* Addison-Wesley Publishing Company, 1994.

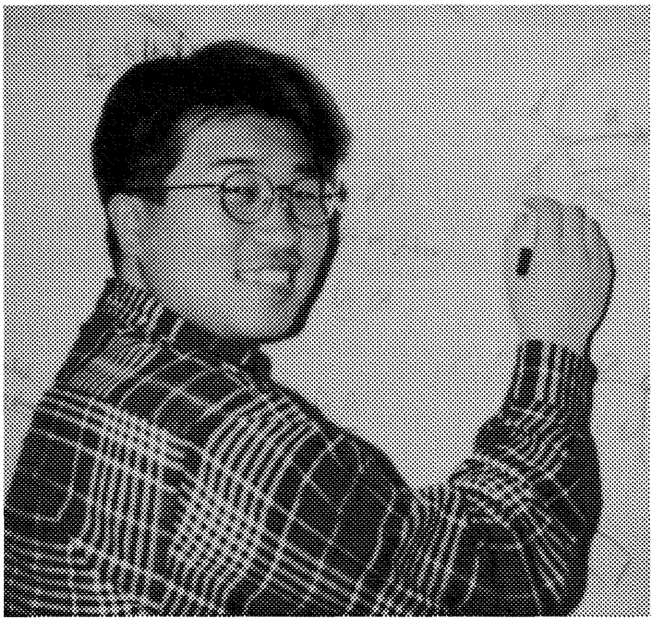
[12] L. Pitt and M. K. Warmuth. The minimum consistent DFA problem cannot be approximated within any polynomial. *J. Assoc. Comput. Mach.*, 40:95–142, 1993.

[13] V. Raghavan. Bounded degree graph inference from walks. *J. Comput. System Sci.*, 49:108–132, 1994.

[14] S. Rudich. Inferring the structure of a markov chain from its output. In *Proc. 26th IEEE Symp. Foundations of Computer Science*, pages 321–326, 1985.

[15] A. Yamaguchi and S. Miyano. Approximating minimum common supertrees for complete k-ary trees. Technical report, Research Institute of Fundamental Information Science, Kyushu University, RIFIS-TR-CS **66**, 1993.

About Authors



Osamu Maruyama was born in Fukuoka on August 14, 1967. He received the B.S. in 1991 from Department of Physics and the M.S. degree in 1993 from Department of Information Systems, Kyushu University He is now studying computational complexity and approximation algorithms.

Satoru Miyano was born in Oita on December 5, 1954. He received the B.S. in 1977, the M.S. degree in 1979 and the Dr. Sci. in 1984 all in Mathematics from Kyushu University. Presently, he is a Professor of Research Institute of Fundamental Information Science, Kyushu University. His present interests include genome informatics, parallel algorithms, computational complexity and computational learning theory.