数理解析研究所講究録
第790巻 1992年 175-193

歩行を実現するグラフ構成問題について

丸山 修 (Osamu Maruyama)†
九州大学大学院総合理工学研究科

Abstract

A walk of an edge-colored undirected graph G is a path which contains all edges in G. We prove results concerning the computational tractability of some problems related to inferring the smallest edge-colored graph from walks. We show an $O(n \log n)$ time algorithm for inferring a tree from a walk which realizes the given walk. If the alphabet of colors is fixed, the algorithm runs in $O(n)$ time. Further, we consider the problem of finding the smallest tree from partial walks, where a partial walk of G is a path in G. We prove that the problem turns to be NP-complete. We also show that inferring the smallest linear chain from partial walks is NP-complete, while the problem of inferring the smallest linear chain from a single walk is known to be solvable in polynomial time.

1. Introduction

Aslam and Rivest [4] considered the problem of inferring the smallest undirected edge-colored graph of bounded degree k consistent with the sequence of edge-colors seen in a walk of the graph. By proving Church-Rosser property of a certain set of rewriting rules, they gave an $O(n^5)$ time algorithm for inferring a linear chain or a cycle from a walk. Raghavan [11] improved their algorithm to an $O(n \log n)$ time algorithm for the case $k = 2$. Further, he showed that the problem is NP-complete for all $k \geq 3$. Angluin [1] and Gold [7] also have discussed a problem closely related to this graph inference. They considered the problem of identifying the smallest finite automaton consistent with given input/output behaviors. This can be regarded as the case that an edge-colored directed graph is to be inferred from strings. They showed that the problem is, in general, NP-complete. Further, Pitt and Warmuth [10] have shown an interesting negative result on approximation algorithms for this problem.

This paper solves the problem for trees. The tree inference from a walk is, for a sequence of edge-colors, to infer the smallest undirected edge-colored tree which produces a walk with the given sequence of edge-colors. We do not put any restriction

†Mailing address: Research Institute of Fundamental Information Science, Kyushu University 33, Fukuoka 812, Japan.
Email: maru@rifis.sci.kyushu-u.ac.jp

on the degree bound. We show an $O(n \log n)$ time algorithm for inferring the tree from a walk which realizes the given walk. When the alphabet of colors is fixed, the algorithm runs in $O(n)$ time.

A partial walk of a graph G is a path in G while a walk of a graph G must contain *all* edges in G. In Section 4 we consider the problem of finding the smallest tree from partial walks. In this problem, the number of partial walks is finite but not bounded. Formally, the problem is, for sequences of edge-colors, to infer the smallest undirected edge-colored tree which allows all paths with the given sequences as labels. We do not put any restriction on the degree bound. In contrast with the case of a single walk, we prove that the problem turns NP-complete. Finally, we consider the problem of inferring a linear chain from partial walks. It has been shown that the linear chain inference from a walk is solvable in polynomial time [4, 11]. We show, however, that the problem of linear chain inference from partial walks is also NP-complete.

This paper is organized as follows. Section 2 contains some necessary notations. Section 3 shows how rewrite rules can be used to find the smallest tree consistent with a given walk. Finally, Section 4 concentrates on the problem of finding the smallest tree and linear chain consistent with a finite number of sequences of edge-colors.

2. Preliminaries

Let Σ be a finite alphabet. We call an element in Σ a *color*. We define an undirected edge-colored graph as $G = (V, E, c)$ with a vertex set V, an edge set E and an edge-coloring $c : E \to \Sigma$. Hereafter a graph means an undirected edge-colored graph without any notice.

A *linear chain* is an undirected edge-colored graph $l = (V, E, c)$ with $V = \{v_1, v_2, \ldots, v_n\}$ and $E = \{\{v_1, v_2\}, \{v_2, v_3\}, \ldots, \{v_{n-1}, v_n\}\}$. We denote the sequence of edge-colors $c(v_1, v_2)$, $c(v_2, v_3), \ldots, c(v_{n-1}, v_n)$ by a string $label(l) \in \Sigma^+$. Let $\Gamma_{\text{lc}}, \Gamma_{\text{tree}}$ and $\Gamma_{k\text{-dg}}$ be classes of linear chains, trees and graph with degree at most k, respectively.

A *partial walk* w on a graph G is a path in G. If a partial walk contains all edges of G, it is called a *walk*. The *scene* in a partial walk w, denoted $scene(w)$, is defined as the sequence of colors of the edges traversed in the walk. For a finite set R of strings from Σ^+ and a graph G, we say that G *realizes all partial walks for* R if G allows a partial walk w_x with $scene(w_x) = x$ for all $x \in R$.

Let $\to$ be a binary relation on a set S, i.e., a subset of $S \times S$. We denote $(x, y) \in \to$ by $x \to y$. The identity relation is $\imath = \{(x, x) | \ x \in S\}$. The union of two relations $\to$ and $\imath$ is denoted by $\xrightarrow{\epsilon}$. The composition of two binary relation is defined in a usual way. The reflexive and transitive closure of $\to$ is denoted by $\xrightarrow{*}$. An element $x \in S$ is called *irreducible* if there is no $y \in S$ with $x \to y$. If $x \xrightarrow{*} y$ and y is irreducible, then we say that y is a *normal form* of x, denoted $\widehat{x}$. A binary relation $\to$ is *noetherian* if there is no infinite sequence $x_1 \to x_2 \to \cdots \to x_n \to \cdots$. A binary relation $\to$ is *confluent* if $w \xrightarrow{*} x$ and $w \xrightarrow{*} y$ imply that there is z with $x \xrightarrow{*} z$ and $y \xrightarrow{*} z$ for any w, x, y. A binary relation $\to$ is *locally confluent* if $w \to x$ and $w \to y$ imply that

there is z with $x \xrightarrow{*} z$ and $y \xrightarrow{*} z$ for any w, x, y. Let $|x|$ denote the size of x. A binary relation $\rightarrow$ is *strictly decreasing* with respect to the size $|x|$ if $x \rightarrow y$ implies $|y| < |x|$ for any x, y.

The following results are well known [8].

Proposition 1.
(1) *For a noetherian relation, it is confluent if and only if it is locally confluent.*
(2) *If a binary relation is confluent, then a normal form of any element, if it exists, is unique.*

Proposition 2. *If a binary relation $\rightarrow$ is confluent and strictly decreasing, then $x \xrightarrow{*} y$ implies $|y| \geq |\hat{x}|$ for any x, y.*

3. Smallest Graph from a Walk

In this section, we consider the problem of constructing from a given string $x \in \Sigma^+$ the smallest tree which allows a walk w with $scene(w) = x$.

Example 1. The graph in Fig. 1 is the smallest tree which produces a walk with the scene `aaabccbbccbdeefff`.

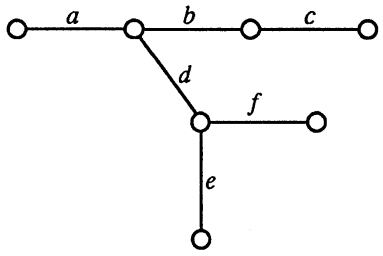


Figure 1: The smallest tree which allows a walk with scene `aaabccbbccbdeefff`.

Definition. Let Γ be a class of graphs.

ISGW(Γ) (Inferring the Smallest Graph from a Walk)
INSTANCE: A finite alphabet Σ and $x \in \Sigma^+$.
PROBLEM: Find a graph $G = (V, E, c) \in \Gamma$ such that G allows a walk w with $scene(w) = x$ and $|E|$ is the smallest.

Proposition 3 (Aslam and Rivest[4]). **ISGW**($\Gamma_{2\text{-dg}}$) is solvable in $O(n^5)$ time.

Proposition 4 (Raghavan[11]). **ISGW**($\Gamma_{2\text{-dg}}$) is solvable in $O(n \log n)$ time.

Definition. Let Γ be a class of graphs.

V-SGW(Γ) (Vertex-Smallest Graph from a Walk)
INSTANCE: A finite alphabet Σ, $x \in \Sigma^+$ and a positive integer K.
QUESTION: Is there a graph $(V, E, c) \in \Gamma$ which allows a walk w with $scene(w) = x$ and $|V| \leq K$?

Proposition 5 (Raghavan[11]). For $k \geq 3$, **V-SGW**($\Gamma_{k\text{-dg}}$) is **NP**-complete.

The main result in this section is the following theorem:

Theorem 1. **ISGW**(Γ_{tree}) is solvable in $O(n \log n)$ time. Furthermore, if Σ is a fixed finite alphabet, the problem is solvable in $O(n)$ time.

Let $g_i = (V_i, E_i, c_i)$ $(1 \leq i \leq n)$ be subgraphs of $\mathcal{G} = (\mathcal{V}, \mathcal{E}, c)$ with $c : \mathcal{E} \to \Sigma$. The *uniongraph* of $g_1, g_2, \ldots, g_n$, denoted $union(g_1, g_2, \ldots, g_n) = (V, E, c')$, is the graph defined as follows:

(1) $V = \bigcup_{1 \leq i \leq n} V_i$.

(2) $E = \bigcup_{1 \leq i \leq n} E_i$.

(3) $c' : E \to \Sigma$ is the restriction of $c : \mathcal{E} \to \Sigma$ to E.

The following lemma is straightforward and its proof is omitted.

Lemma 1. Let w be a walk on a tree t. Then there exist linear chains l_i $(1 \leq i \leq n)$ in t satisfying the following conditions:

(1) The rightmost edge of l_j is the mirror image of the leftmost edge of l_{j+1}, i.e., these edges are the same and the rightmost vertex of l_j is the same as the leftmost vertex of l_{j+1}, for $1 \leq j \leq n-1$ (see Fig. 2).

(2) $union(l_1, l_2, \ldots, l_{n-1}, l_n) = t$.

(3) $scene(w) = label(l_1)\ label(l_2) \cdots label(l_n)$.

Further, these linear chains $l_1, \ldots, l_n$ are unique for w and t.

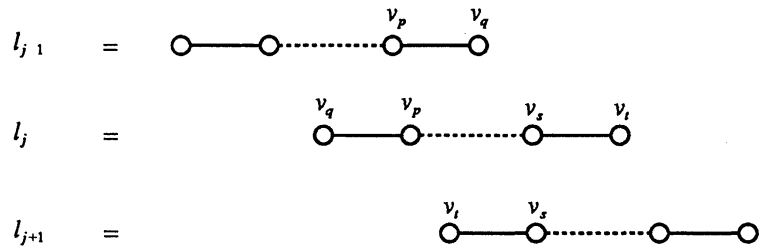
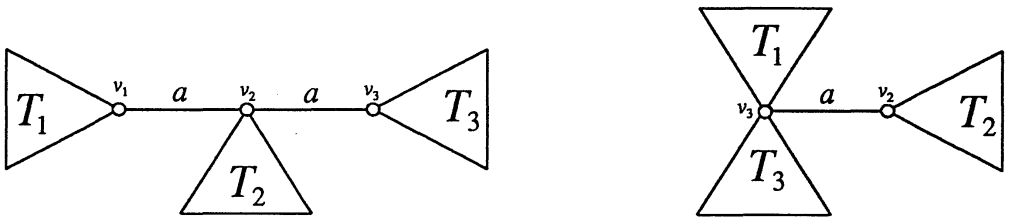


Figure 2: v_p, v_q, v_s and v_t are vertices.



(a) t_1 (b) t_2

Figure 3: T_1, T_2 and T_3 are arbitrary trees and a is an arbitrary color in Σ.

Definition. Let t_1 be a tree of the form shown in Fig. 3 (a) consisting of subtrees T_1, T_2 and T_3 with roots v_1, v_2, v_3, respectively, which are joined by edges $\{v_1, v_2\}$ and $\{v_2, v_3\}$ with the same color $a \in \Sigma$. Then let t_2 be the tree shown in Fig. 3 (b) obtained from t_1 by identifying v_1 with v_3 together with the adjacent edges. We say that t_2 is an *edge-folding* of t_1. The relation $\rightarrow_F$ consists of the pairs (t_1, t_2) such that t_1 is a tree of the form in Fig. 3 (a) and t_2 is an edge-folding of t_1.

For a tree t, the size $|t|$ is the number of edges in t. Clearly, $\rightarrow_F$ is strictly decreasing with respect to $|t|$ since $t_1 \rightarrow_F t_2$ implies $|t_2| < |t_1|$.

Definition. Let $g_1 = (V_1, E_1, c_1)$ and $g_2 = (V_2, E_2, c_2)$ be graphs with $v_1 \in V_1$ and $v_2 \in V_2$. Let $g_1' = (V_1', E_1', c_1')$ and $g_2' = (V_2', E_2', c_2')$ be disjoint copies of g_1 and g_2 with $V_1' \cap V_2' = \emptyset$, respectively. The graph $\langle g_1 \triangleright v_1 \circ v_2 \triangleleft g_2 \rangle$ is the graph obtained from g_1' and g_2' by identifying v_1' with v_2', where v_1' and v_2' are vertices corresponding to v_1 and v_2, respectively.

Lemma 2. *Let t be a tree and x be a string in Σ^+. The following statements are equivalent:*

(a) *The tree t allows a walk w with $scene(w) = x$.*

(b) *The linear chain l with $label(l) = x$ satisfies $l \xrightarrow{*}_F t$.*

Proof. (a) $\Rightarrow$ (b) : Let w be a walk on t with $scene(w) = x$. Then there exist linear chains $l_1, l_2, \ldots, l_n$ in t satisfying (1)-(3) of Lemma 1. For $1 \leq k \leq n$, let $l_1 \cdots l_k$ be the linear chain which concatenates $l_1, \ldots, l_k$ by identifying the rightmost vertex of l_i with the leftmost vertex of l_{i+1} for $1 \leq i < k-1$. We show by induction on k that $l_1 \cdots l_k \xrightarrow{*}_F union(l_1, \ldots, l_k)$ holds for $1 \leq k \leq n$. The case of $k = 1$ is trivial. Assume $l_1 \cdots l_{k-1} \xrightarrow{*}_F union(l_1, \ldots, l_{k-1})$. Let v_1 be the rightmost vertex of l_{k-1} and v_2 be the leftmost vertex of l_k. By definition, $l_1 \cdots l_{k-1} l_k = \langle l_1 \cdots l_{k-1} \triangleright v_1 \circ v_2 \triangleleft l_k \rangle$. Since $l_1 \cdots l_{k-1} \xrightarrow{*}_F union(l_1, \ldots, l_{k-1})$, we can see (Fig. 4)

$$\langle union(l_1, \ldots, l_{k-1}) \triangleright v_1 \circ v_2 \triangleleft l_k \rangle \xrightarrow{*}_F union(l_1, \ldots, l_{k-1}, l_k).$$

Thus $l_1 \cdots l_n \xrightarrow{*}_F union(l_1, l_2, \ldots, l_n)$. By (2) of Lemma 1, $union(l_1, l_2, \ldots, l_n) = t$. Hence $l_1 \cdots l_n \xrightarrow{*}_F t$. By (3) of Lemma 1, $label(l_1)\ label(l_2) \cdots label(l_n) = scene(w)$. Since $label(l_1)\ label(l_2) \cdots label(l_n) = label(l_1 \cdots l_n)$, the linear chain $l = l_1 \cdots l_n$ satisfies (b).

(b) $\Rightarrow$ (a) : Let t_1 and t_2 be trees such that $t_1 \rightarrow_F t_2$ as shown in Fig. 3 (a) and (b), respectively. It suffices to show that for any walk w_1 on t_1 there is a walk w_2 on t_2 with $scene(w_2) = scene(w_1)$. There are several cases to consider. For example, if w_1 contains a path $v_1 \overset{a}{-} v_2 \overset{a}{-} v_3$ in t_1, then we replaced it by a path $v_3 \overset{a}{-} v_2 \overset{a}{-} v_3$ in t_2. This replacement does not change the scene of walks. Fig. 5 shows the replacements in all cases. Let l be a linear chain with $label(l) = x$ such that $l \xrightarrow{*}_F t$. Clearly, there is a walk w on l with $scene(w) = x$. Therefore by the above argument there is a walk w' on t with $scene(w') = scene(w) = x$. □

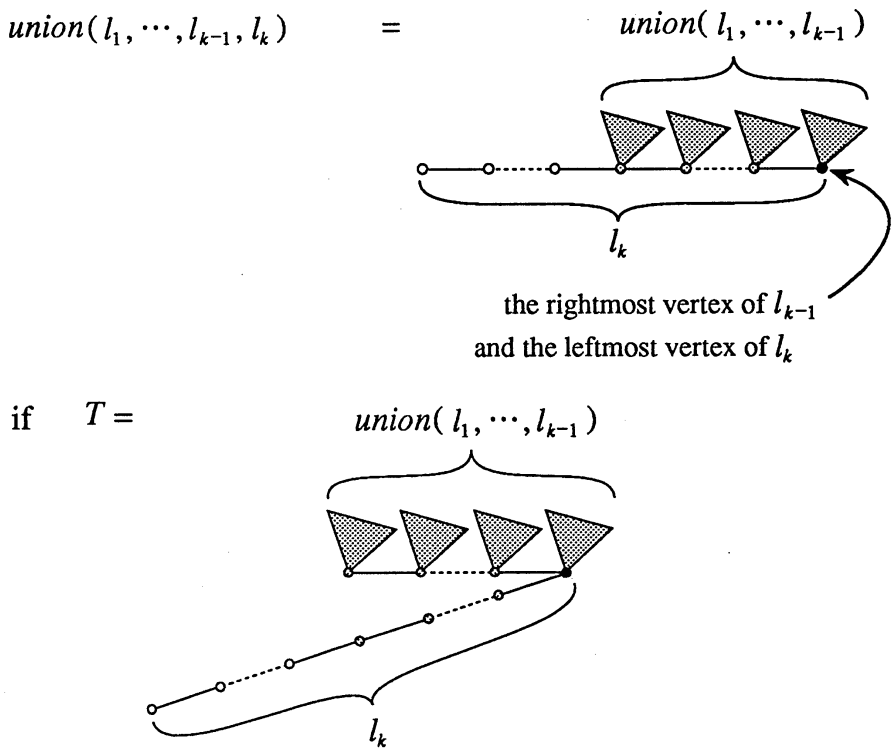


Figure 4: The relation of $union(l_1, \ldots, l_{k-1})$ and $union(l_1, \ldots, l_{k-1}, l_k)$

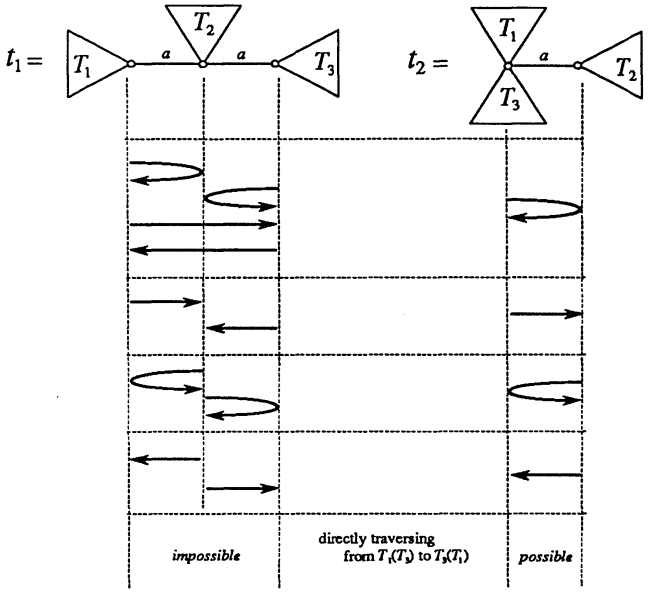


Figure 5: T_1, T_2 and T_3 are arbitrary trees and a is an arbitrary color in Σ.

Lemma 3. *The binary relation $\rightarrow_F$ is confluent.*

Proof. Since $\rightarrow_F$ is strictly decreasing, by Proposition 1 (1) we need only to show that $\rightarrow_F$ is locally confluent. For $\rightarrow_F$ to be locally confluent, we must have that if $t \rightarrow_F t_1$ and $t \rightarrow_F t_2$ then there exists some tree t' with $t_1 \stackrel{*}{\rightarrow}_F t'$ and $t_2 \stackrel{*}{\rightarrow}_F t'$.

If $t \rightarrow_F t_1$, let $\{v_1, v_2\}$ and $\{v_2, v_3\}$ be the edges in t that are folded in the derivation $t \rightarrow_F t_1$. For $t \rightarrow_F t_2$, let $\{v_4, v_5\}$ and $\{v_5, v_6\}$ be the folded edges in t. We must consider the following three cases. In the case that $\{\{v_1, v_2\}, \{v_2, v_3\}\} \cap \{\{v_4, v_5\}, \{v_5, v_6\}\} = \emptyset$, it is trivial that there exists some tree t' with $t_1 \stackrel{*}{\rightarrow}_F t'$ and $t_2 \stackrel{*}{\rightarrow}_F t'$. In the case that $\{\{v_1, v_2\}, \{v_2, v_3\}\} = \{\{v_4, v_5\}, \{v_5, v_6\}\}$, it is also trivial. When $|\{\{v_1, v_2\}, \{v_2, v_3\}\} \cap \{\{v_4, v_5\}, \{v_5, v_6\}\}| = 1$, without loss of generality, we may assume that $v_2 = v_4$ and $v_3 = v_5$. We can draw t as a tree of the form shown in Fig. 6(a). Then t_1 and t_2 are the trees shown in Fig. 6(b) and (c), respectively. Let t' is the tree shown in Fig. 6 (d). Then we can easily see that $t_1 \rightarrow_F t'$ and $t_2 \rightarrow_F t'$. Thus we have shown that $\rightarrow_F$ is confluent. □

Lemma 4. *For a string x of colors, the normal form $\widehat{l}$ of the linear chain l with $label(l) = x$ is the smallest tree that allows a walk w with $scene(w) = x$.*

Proof. Since $l \stackrel{*}{\rightarrow}_F \widehat{l}$ and l allows a walk with scene x, it follows from Lemma 2 that $\widehat{l}$ also allows a walk with the same scene x. Let t be a tree which allows a walk with scene x. By Lemma 2, the linear chain l satisfies $l \stackrel{*}{\rightarrow}_F t$. Since the binary relation

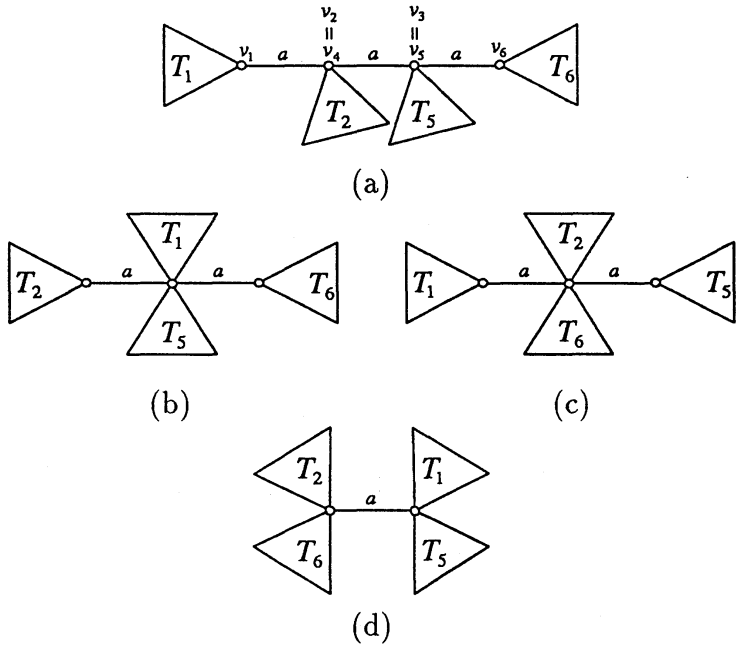


Figure 6: T_1, T_2, T_5 and T_6 are arbitrary trees and a is an arbitrary color in Σ.

$\rightarrow_F$ is confluent by Lemma 3 and strictly decreasing, we see from Proposition 2 that $|t| \geq |\widehat{l}|$. Hence $\widehat{l}$ is the smallest tree that allows a walk w with $scene(w) = x$. □

Input: string $x := x_1 x_2 \cdots x_n$ in Σ^+
Output: tree $t = (V, E, c)$ with a coloring $c : E \to \Sigma$
Procedure:

```
V := {v0}; E := ∅;
v := v0
for i := 1 to n
    if there is a vertex v' ∈ V with {v, v'} ∈ E and c({v, v'}) = x_i
                                              /* see Fig. 7 (a) */
        then v := v'                          /* see Fig. 7 (b) */
        else
            Let u be a new vertex;
            V := V ∪ {u};
            E := E ∪ {{u, v}};
            c({u, v}) := x_i;                 /* see Fig. 7 (c) */
            v := u
    endif
end
```

Algorithm 1: Algorithm for obtaining the normal form of x.

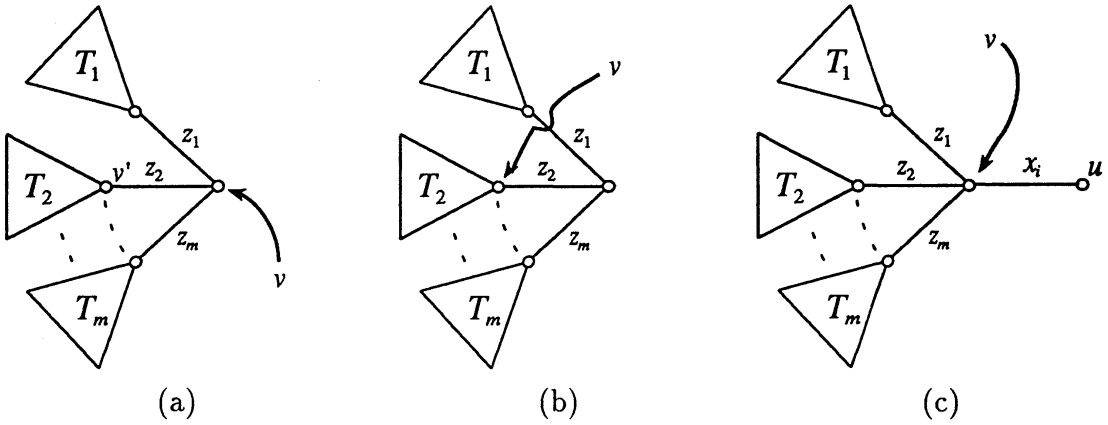


Figure 7: $T_1, T_2, \ldots, T_m$ are arbitrary trees and $z_1, z_2, \ldots, z_m$ are arbitrary distinct colors.

Lemma 5. *On input $x = x_1 x_2 \cdots x_n (x_1, x_2, \ldots, x_n \in \Sigma)$, Algorithm 1 produces the tree in the normal form allowing a walk with scene x.*

Proof. Let $t_i = (V_i, E_i, c_i)$ and v_i be the resulting tree and the content of v just after the ith iteration of the **for**-loop of Algorithm 1, respectively. Let $l_{i,j}$ be the linear chain with $label(l_{i,j}) = x_i x_{i+1} \cdots x_j$ and $lmv(l_{i,j})$ be the leftmost vertex of $l_{i,j}$ for

$1 \leq i \leq j \leq n$. In particular, $l = l_{1,n}$. Clearly, $t_1 = l_{1,1}$ and $\langle t_{i-1} \triangleright v_{i-1} \circ lmv(l_{i,i}) \triangleleft l_{i,i} \rangle \xrightarrow{\epsilon}_F t_i$ for any $i > 1$. Thus, $l = \langle t_1 \triangleright v_1 \circ lmv(l_{2,n}) \triangleleft l_{2,n} \rangle \xrightarrow{\epsilon}_F \langle t_2 \triangleright v_2 \circ lmv(l_{3,n}) \triangleleft l_{3,n} \rangle \xrightarrow{\epsilon}_F \cdots \xrightarrow{\epsilon}_F t_n$. Hence $l \xrightarrow{*}_F t_n$.

We show by induction that t_i is irreducible for all $i = 1, \ldots, n$. Clearly, $t_1 = l_{1,1}$ is irreducible. Assume that that t_{i-1} is irreducible. Consider the **if**-condition of Algorithm 1 that checks whether $x_i = z_j$ for some $j = 1, \ldots, m$. If it is true, then $t_i = t_{i-1}$. Otherwise, t_i is of the form as in Fig. 7 (c). Since t_{i-1} is irreducible, t_{i-1} does not contain a subgraph of the form in Fig. 8 for any $a \in \Sigma$. Furthermore, since $x_i \neq z_j$ for all $j = 1, \ldots, m$, no subgraph of the form in Fig. 8 can appear in t_i. Hence t_i is also irreducible.

Thus, we have shown that $l \xrightarrow{*}_F t_n$ and t_n is irreducible which means that t is the normal form of l. By Lemma 2, t_n allows a walk with scene x since the linear chain l does. □

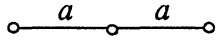


Figure 8:

It is not hard to implement Algorithm 1 so that it runs in $O(n \log n)$ time using a balanced binary tree for keeping colors at each vertex. If Σ is a fixed finite alphabet, it can be implemented in $O(n)$ time.

4. Smallest Graph from Partial Walks

Instead of dealing with a single walk, we consider in this section the problem of finding the smallest tree from a finite number of partial walks.

Example 2. The graph in Fig. 9 is the smallest linear chain which allows partial walks with scenes `abbaabcdeedc`, `cdebccbeddebbe`, `ecbeebccbbcee`.

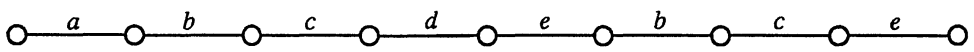


Figure 9:

Definition. Let Γ be a class of graphs.

SGPWs(Γ) (Smallest Graph from Partial Walks)

INSTANCE: A finite alphabet Σ, a finite set R of strings from Σ^+ and a positive integer K.

QUESTION: Is there a graph $(V, E, c) \in \Gamma$ which allows a partial walk w_x with $scene(w_x) = x$ for each $x \in R$ with $|E| \leq K$?

4.1. Inferring a Tree from Partial Walks

We prove the following theorem:

Theorem 2. **SGPWs**(Γ_{tree}) is **NP**-complete.

Proof. The vertex cover problem (VC) [6] is to decide if, given a graph $G = (V, E)$ and a positive integer K, there is a vertex cover of size K or less for G, that is, a subset $V' \subseteq V$ such that $|V'| \leq K$ and, for each edge $\{u, v\} \in E$, at least one of u and v belongs to V'.

We will reduce VC to **SGPWs**(Γ_{tree}). Let $G = (V, E)$ and K be a graph and an integer which forms an instance in VC, where $V = \{v_1, ..., v_n\}$ and $E = \{e_1, ..., e_m\}$.

From G and K, we define a finite alphabet Σ, a finite set R of strings from Σ^+ and a positive integer K'. The finite alphabet Σ is defined as

$$\begin{aligned} \Sigma = \ & \{w_i \mid 1 \leq i \leq W\} && \cup \\ & \{v_i^j \mid 1 \leq i \leq n, 1 \leq j \leq N\} && \cup \\ & \{e_k^j \mid 1 \leq k \leq m, 1 \leq j \leq S\} && \cup \\ & \{\theta\}, \end{aligned}$$

where integers W, N and S shall be defined later.

Next, we define the finite set R of strings from Σ^+. In order to define R, we introduce the following notations for strings.

$$\begin{aligned} [w] &= w_1 w_2 \cdots w_W && : \textit{weight-block} \\ [e_k] &= e_k^1 e_k^2 \cdots e_k^S \quad \text{for each edge } e_k && : \textit{edge-block} \\ [v_i] &= v_i^1 v_i^2 \cdots v_i^N \quad \text{for each vertex } v_i && : \textit{vertex-block} \\ [\theta] &= \theta && : \textit{θ-block} \end{aligned}$$

R consists of the following strings:

Base string: $[w]\,[v_1]\,[w]\,[v_2]\,[w] \cdots [w]\,[v_n]$

Edge strings: $[e_k]\,[v_i]\,[\theta]\,[v_j]^R\,[e_k]^R$ for each edge $e_k = \{v_i, v_j\} \in E$

Connector strings: $[w]\,[e_k]^R$ for each edge $e_k \in E$

We define $N = K + 1$, $S = (|V| + |E| + 1)(K + 1)$, and $W = 2|E|(|V| + |E| + 1)(K + 1) + (|V| + |E|)(K + 1) + K + 1$. Then the following inequalities hold:

(1) $W > 2|E|S + (|V| + |E|)N + K$.

(2) $S > (|V| + |E|)N + K$.

(3) $N > K$.

Finally, let $K' = |V|W + 2|E|S + (|V| + |E|)N + K$. This transformation can be done in polynomial time.

We claim that G has a vertex cover of the size of at most K if and only if there is a tree $t = (V_t, E_t, c_t) \in \Gamma_{\text{tree}}$ with $|E_t| \leq K'$ which allows a partial walk w_x with $scene(w_x) = x$ for each $x \in R$.

Suppose that G has a vertex cover V' with $|V'| \leq K$. We shall define from V' a tree $t = (V_t, E_t, c_t)$ with $|E_t| \leq K'$ which realizes all partial walks for R. Since any $x \in R$ is irreducible with respect to the binary relation $\rightarrow_F$, we can see from Lemma 2 that the linear chain l_x with $label(l_x) = x$ is a unique tree which allows a *walk* w_x with $scene(w_x) = x$. Therefore a tree which realizes all partial walks for R must contain such linear chains l_x as subgraphs for all $x \in R$. Hereafter a block or string x means the linear chain l_x with $label(l_x) = x$ without any notice. For graphical representation, we can represent the linear chain $l_{[w]}$ for a weight-block $[w]$ by $\xrightarrow{[w]}$ since the symbols defining $[w]$ are mutually distinct. Similarly, for an edge-block $[e_k]$ and a vertex-block $[v_i]$, we represent the corresponding linear chains by $\xrightarrow{[e_k]}$ and $\xrightarrow{[v_i]}$, respectively. On the other hand, we represent the linear chain $l_{[\theta]}$ for a θ-block $[\theta]$ by $\xleftrightarrow{[\theta]}$ since it is symmetric.

Initially, we are given the collection $l(R)$ of the linear chains l_x for all $x \in R$. We construct the tree t by overlapping these linear chains each other.

First, we consider the connector strings and the edge strings. For each edge $e_k = \{v_i, v_j\} \in E$, either v_i or v_j is in V'. If v_i is in V', we make a tree by overlapping the edge-block $[e_k]$ of the connector string $[w]\ [e_k]^R$ with the $[e_k]$ of the edge string $[e_k]\ [v_i]\ [\theta]\ [v_j]^R\ [e_k]^R$ which is adjacent to the vertex-block $[v_i]$ as shown in Fig. 10. These two edge-blocks match exactly by reversing the direction appropriately. We denote the resulting tree by $t(v_i, e_k)$ which means that the endpoint v_i is chosen from e_k in V'. If v_j is in V', we can define $t(v_j, e_k)$ similarly. We construct either $t(v_i, e_k)$ or $t(v_j, e_k)$ for e_k, and call the tree the *endpoint selection tree* for e_k.

Next, we consider the endpoint selection trees for edges and the base string $[w]\ [v_1]\ [w]\ [v_2]\ [w] \cdots [w]\ [v_n]$. For an edge $e_k = \{v_i, v_j\}$, let the endpoint selection tree be $t(v_i, e_k)$. Then we overlap the consecutive blocks $[w]\ [v_i]$ of $t(v_i, e_k)$ with the consecutive blocks $[w]\ [v_i]$ of the base string $[w]\ [v_1]\ [w]\ [v_2]\ [w] \cdots [w]\ [v_n]$. The resulting tree looks like Fig.11. We call the location of the base string between $[w]$ and $[v_i]$ the *vertex-selection point.*

Finally, for each vertex-selection point, we overlap the θ-blocks of the endpoint section trees which share the same vertex-selection point each other as shown in Fig.12. Then the resulting tree gives the tree t to be constructed. It is obvious from the construction of t that t realizes all partial walks for R.

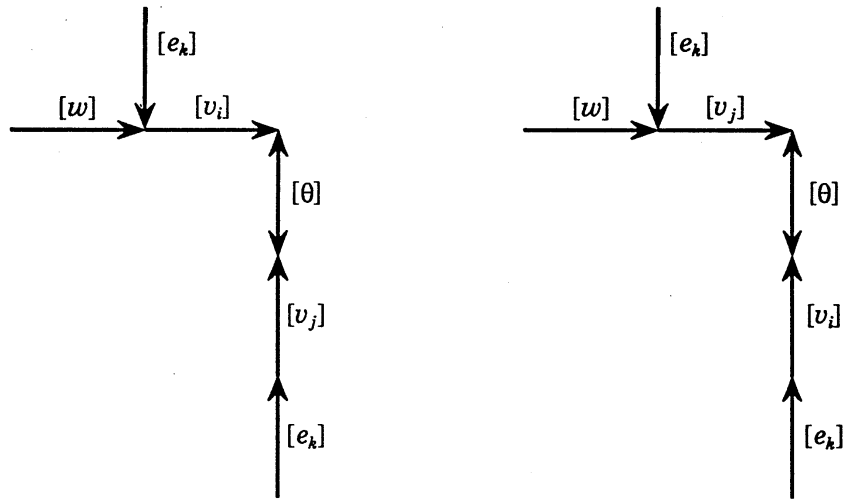


Figure 10: The endpoint selection trees for e_k

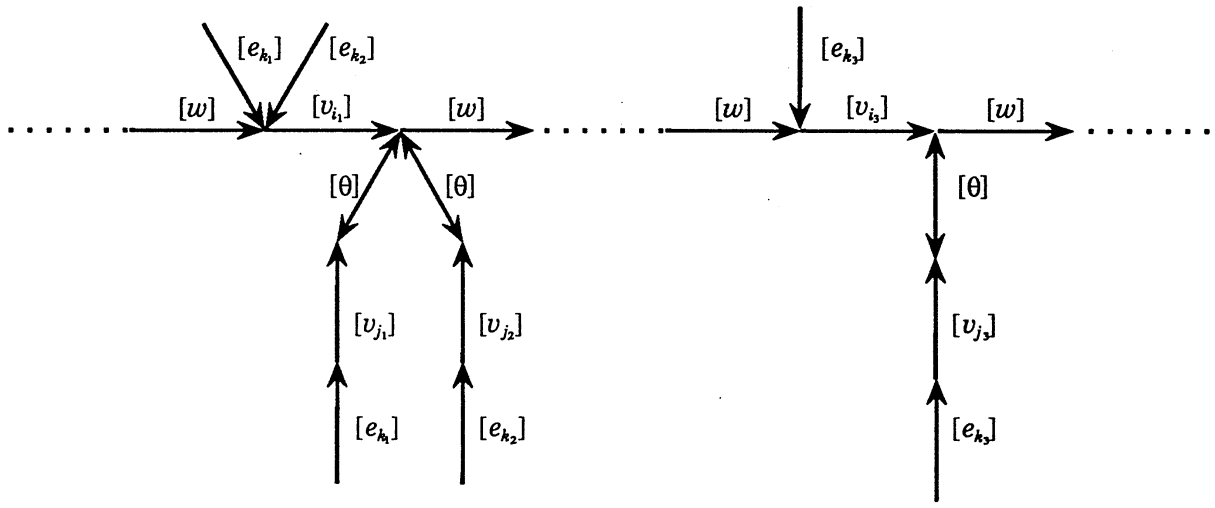


Figure 11: Overlapping of the endpoint selection trees with the base string

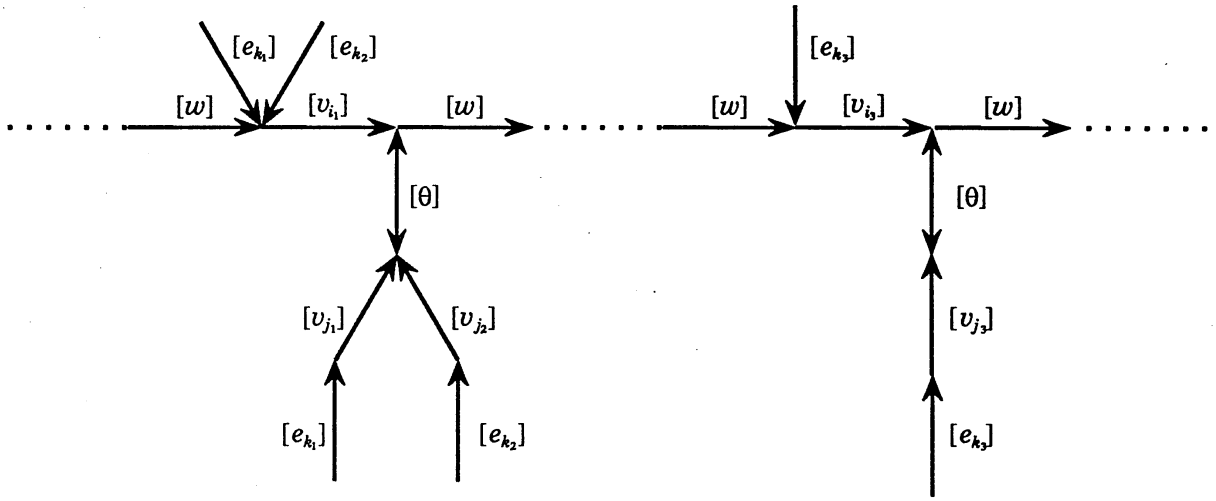


Figure 12: Tree realizing all partial walks for R

The number of vertex-selection points which are adjacent to θ-blocks is at most K since $|V'| \leq K$. Then it can be easily checked that t contains at most K' edges.

Conversely, let $t = (V_t, E_t, c_t)$ be a tree with $|E_t| \leq K'$ which realized all partial walks for R. Such tree t can be obtained by overlapping the linear chains in $l(R)$. Without loss of generality, we may assume that t is one of the smallest such trees. Since the collection $l(R)$ of linear chains for R contains $(|V|+|E|)W+3|E|S+(|V|+2|E|)N+|E|$ edges, at least $|E|W+|E|S+|E|N+(|E|-K)$ edges must be eliminated from $l(R)$ by overlapping edges in $l(R)$.

In overlapping these linear chains, it should be noticed that two blocks (weight-blocks, edge-blocks, vertex-blocks, or θ-blocks) of the linear chains must overlap completely in t, if they do, since t is assumed to be the smallest one. Note also that distinct blocks do not share any symbols.

1. We consider the connector strings for edges and the base string. We show that every connector string must overlap with the base string by the part of weight blocks $[w]$ as shown in Fig.13. If there is a connector string $[w]\ [e_k]^R$ whose weight-block $[w]$ is not overlapped with the base string $[w]\ [v_1]\ [w]\ [v_2]\ [w]\ \cdots\ [w]\ [v_n]$ in the tree t, then t contains at least $|V| \cdot W + W$ edges. By (1), W is chosen so that $W > 2|E|S + (|V| + |E|)N + K$. This contradicts the assumption that t contains at most K' edges. Let t' be the resulting graph formed from the base string and the connector strings. At this point, $|V| \cdot W$ edges are eliminated from $l(R)$ by overlapping.

2. We consider the edge strings.

Claim 1. Every edge string must overlap with t' by the part of edge-blocks.

Proof. For two edges $e_{k_1} = \{v_{i_1}, v_{j_1}\}$, $e_{k_2} = \{v_{i_2}, v_{j_2}\}$, the edge-blocks $[e_{k_1}]$ of the edge string $[e_{k_1}]\ [v_{i_1}]\ [\theta]\ [v_{j_1}]^R\ [e_{k_1}]^R$ cannot overlap with the edge-blocks $[e_{k_2}]$

Figure 13: Overlapping of the connector strings with the base string

of the edge string $[e_{k_2}]\ [v_{i_2}]\ [\theta]\ [v_{j_2}]^R\ [e_{k_2}]^R$ since $[e_{k_1}]$ and $[e_{k_2}]$ have no symbol in common. If there is an edge string $[e_k]\ [v_i]\ [\theta]\ [v_j]^R\ [e_k]^R$ none of whose edge-blocks is not overlapped with the edge-block $[e_k]$ of t', then t must contain at least $|V| \cdot W + (2|E|+1) \cdot S$ edges. However, by (2), S is chosen so that $S > (|V|+|E|)N + K$. This implies that t contains more than K' edges, a contradiction. Therefore one of the edge-blocks of each edge string is overlapped with the same edge-block of t' as shown in Fig.14, where the part of an edge string other than the overlapped edge-block is drawn out temporally. At this point, $|V| \cdot W + |E| \cdot S$ edges are eliminated from $l(R)$.

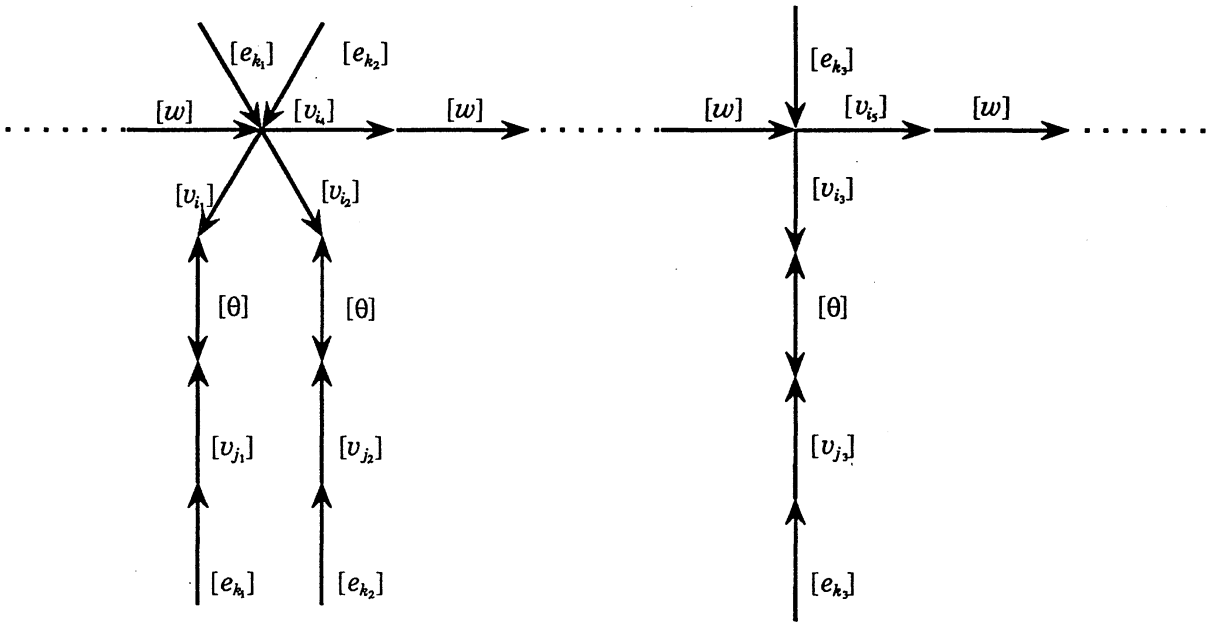


Figure 14: Overlapping of the edge-blocks of the edge strings with the base string

Claim 2. Let $[e_k]\ [v_i]\ [\theta]\ [v_j]^R\ [e_k]^R$ be an edge string. If the left (right) edge-block $[e_k]$ is overlapped with t', then the vertex-block $[v_i]$ ($[v_j]$) of the edge string must be overlapped with the same vertex-block of t'.

Proof. By an argument similar to Claim 1, we can show the above claim by using the inequality (3) $N \geq K$. At this point, $|V| \cdot W + |E| \cdot S + |E| \cdot N$ edges are eliminated from $l(R)$.

Claim 3. The number of θ-blocks in t is at most K.

Proof. Since t is assumed to be the smallest one, all θ-blocks adjacent to each vertex-selection point are overlapped into one θ-block as in Fig.12. Since t contains at most $|V|W + 2|E|S + (|V| + |E|)N + K$ edges, we see that the number of the θ-blocks must be at most K.

It should be clear that the set V' of vertices v_i corresponding to the vertex-selection points in $[w]$ $[v_i]$ of t which are adjacent to θ-blocks gives a vertex cover of G whose size is shown at most K.

Thus the problem of inferring a tree from partial walks is NP-hard. Clearly, the problem is in NP. □

4.2. Inferring a Linear Chain from Partial Walks

Theorem 3. **SGPWs**(Γ_{lc}) is NP-complete.

Proof. We show that this problem is a special case of the shortest common superstring problem [5, 6, 9], which is to decide if, given a finite set R of strings from Σ^* and a positive integer K, there is a string $s \in \Sigma^*$ with $|s| \leq K$ such that each string $x \in R$ is a substring of s.

For a string $x = x_1x_2 \cdots x_n$ $(x_i \in \Sigma)$, we make a new string $\tilde{x} = abx_1abx_2ab \cdots abx_nab$ from x using new symbols a and b not in Σ. Now we define an instance of **SGPWs**(Γ_{lc}) as follows:

(1) $\Sigma' = \Sigma \cup \{a, b\}$ with $a, b \notin \Sigma$.

(2) $R' = \{\tilde{x} \mid x \in R\}$.

(3) $K' = 3K + 2$.

This transformation can be done in polynomial time.

We claim that there is a common superstring s for R with $|s| \leq K$ if and only if there is a linear chain l' with $|label(l')| \leq K'$ which allows a partial walk $w_{\tilde{x}}$ with $scene(w_{\tilde{x}}) = \tilde{x}$ for each $\tilde{x} \in R'$.

Suppose that there is a common superstring s for R with $|s| \leq K$. Let s be $s_1s_2 \cdots s_n$ with $s_i \in \Sigma$, and let l be the linear chain with $label(l) = abs_1abs_2ab \cdots abs_nab$. Then $|label(l)| \leq 3K + 2$. We can easily see that the linear chain l allows a partial walk $w_{\tilde{x}}$ with $scene(w_{\tilde{x}}) = \tilde{x}$ for each $\tilde{x} \in R'$.

We prove the converse. Let $\rightarrow_U$ be the binary relation on Σ'^+ defined as follows:

$$\begin{aligned} \rightarrow_U = \; & \{(pxx^Rxq, pxq) \mid p, q \in \Sigma'^*, x \in \Sigma'^+\} \cup \\ & \{(x^Rxq, xq) \mid q \in \Sigma'^*, x \in \Sigma'^+\} \cup \\ & \{(pxx^R, px) \mid p \in \Sigma'^*, x \in \Sigma'^+\}, \end{aligned}$$

where x^R is the reverse of the string x. It has been shown in [4] that, for any string $y, z \in \Sigma'^+$, $y \xrightarrow{*}_U z$ if and only if the linear chain l with $label(l) = z$ allows a walk w with $scene(w) = y$. Since any $\tilde{x} \in \tilde{R}$ is irreducible with respect to the binary relation $\rightarrow_U$, we can see that the linear chain l with $label(l) = \tilde{x}$ is the only linear chain which allows a walk w with $scene(w) = \tilde{x}$.

Let l be a linear chain with $|label(l)| \leq K'$ which allows a partial walk $w_{\tilde{x}}$ with $scene(w_{\tilde{x}}) = \tilde{x}$ for each $\tilde{x}$ in R'. Then the string $s' = label(l)$ satisfies $|s'| \leq K'$ and $\tilde{x}$ or $\tilde{x}^R$ is a substring of s' for all $\tilde{x}$ in R'. Without loss of generality, we may assume that l is one of the shortest such linear chains and s' begins with ab.

For strings $\tilde{x} = abx_1abx_2ab\cdots abx_mab$ and $\tilde{y} = aby_1aby_2ab\cdots aby_nab$, we can see that $\tilde{x}$ and $\tilde{y}^R$ can overlap maximally with the form either $ab\tilde{x}'aba\tilde{y}'^Rba$ or $ba\tilde{y}'^Rbab\tilde{x}'ab$. On the other hand, $\tilde{x}$ and $\tilde{y}$ can overlap at least as $abx_1abx_2ab\cdots abx_maby_1aby_2ab\cdots aby_nab$. This overlapped string is one symbol shorter than $ab\tilde{x}'aba\tilde{y}'^Rba$ and $ba\tilde{y}'^Rbab\tilde{x}'ab$. By this observation, we can conclude that s' is of the form $s' = abs_1abs_2ab\cdots abs_tab$ with $s_i \in \Sigma$. Therefore s' contains all strings $\tilde{x}$ in R'. Hence $s = s_1s_2\cdots s_t$ is a common superstring for R. Further, since $|s'| = 3|s| + 2 \leq K' = 3K + 2$, we see $|s| \leq K$. Thus $\mathbf{SGPWs}(\Gamma_{lc})$ is NP-hard. Clearly, the problem is in NP. □

5. Toward Bioinformatics

Our motivation of this work comes from knowledge acquisition from amino acid sequences of proteins and DNA sequences which are strings from 20 symbols and four symbols (A,T,C,G), respectively. One of the important topics in Molecular Biology is to develop systematic methods which discover new knowledge about these sequences. The approach in this paper is based on the following principle: The smallest hypothesis which explains given data exhibits the knowledge. With this principle, we have developed two methods for amino acid sequences of proteins with quite successful results [2, 3].

Aslam and Rivest [4] deals with linear chains and cycles. Raghavan [11] considers graphs with degree at most k. This paper established the results for trees. However, there are a various kinds of graph families which are remained for investigation. We believe that graph inference from walks may provide a new method for discovering knowledge in Molecular Biology or in another fields which confront with the problem of knowledge acquisition. We are now in the process of applying our algorithm for trees and the algorithms for linear chains by [4, 11] to various amino acid sequences and DNA sequences.

6. Acknowledgment

The authors would like to thank Ayumi Shinohara for a great amount of helps and suggestions in attacking the problems discussed in this paper. Especially, his suggestion of using the common superstring problem for the linear chain inference from partial walks is appreciated very much.

References

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

[2] S. Arikawa, S. Kuhara, S. Miyano, A. Shinohara and T. Shinohara, A learning algorithm for elementary formal systems and its experiments on identification of transmembrane domains, to appear in Proc. 25th Hawaii International Conference on System Sciences.

[3] S. Arikawa, S. Kuhara, S. Miyano, A. Shinohara and T. Shinohara, Identification of transmembrane domains by decision trees over regular patterns, RIFIS-TR-CS-44, Research Institute of Fundamental Information Science, Kyushu University, August, 1991 (to be presented at 2nd Int. Symposium on Artificial Intelligence and Mathematics, Florida, January, 1992).

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

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

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

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

[8] G. Huet, Confluent reductions: abstract properties and applications to term rewriting systems, *J. Assoc. Comput. Mach.* **27** (1980) 797–821.

[9] M. Li, Towards a DNA sequencing theory, *Proc. 31st IEEE Symposium on Foundations of Computer Science*, 125–134, 1990.

[10] L. Pitt and M.K. Warmuth, The minimum consistent DFA problem cannot be approximated within any polynomial, *Proc. 21st ACM Symposium on Theory of Computing*, 421–432, 1989.

[11] V. Raghavan, Bounded degree graph inference from walks, *Proc. 4th Computational Learning Theory* 354–366, 1991.