数理解析研究所講究録
第 427 巻 1981 年 131-147

ON COMMON SEQUENCE PROBLEMS

Tohru KIKUNO, Noriyoshi YOSHIDA and Shin'ichi WAKABAYASHI
Faculty of Engineering, Hiroshima University

1. Introduction

A common sequence problem [3] is one of the generalized problems of both the string-matching and the string-to-string correction problems [13]. The problem arises in data processing such as comparing two files [2] and in genetics such as studying molecular evolution [8]. Concerning common sequence problem, the following two subproblems are defined: the longest common subsequence problem, shortly, the LCS problem, and the shortest common supersequence problem, shortly, the SCS problem.

In this paper, we show that the LCS problem is solvable in polynomial time for primitive strings, whereas the LCS problem is NP-complete [7] in general. Concerning the SCS problem we give strictly stronger results than the previous one [7] by showing that the yes/no SCS problem is NP-complete for strings whose length is bounded 5 except one string. We also show that the yes/no SCS problem is NP-complete for primitive strings.

Then we present an effective approximation algorithm SCSARG, based on a mathematical model ARG, for the SCS problem. The algorithm SCSARG can be applied to the design of the Printed Wiring Boards, resulting in that both placement and routing are determined almost simultaneously. It implies that we never fail to determine routing if there is no physical restriction such as the area of the board.

2. Preliminaries

We present necessary definitions about common sequences and some decision problems on them [7,14].

Definition 1. Let S be a finite string (or sequence) over an alphabet Σ. S' is a subsequence of S if there exist integers $1 \leq r_1 < r_2 < \ldots < r_s \leq |S|$ such that $S'[i]=S[r_i]$ for each i, $1 \leq i \leq |S'|$, where S[i] denotes i-th character in a string S. We use $S>S'$ to denote that S' is a subsequence of S. Conversely S is a supersequence of S' if S' is a subsequence of S.

Definition 2. Let $R=\{S_1,S_2,\ldots,S_p\}$ be a set of p strings over Σ, where $p \geq 2$. A common subsequence of R is a string $\tilde{S}$ such that $S_i>\tilde{S}$ for each i, $1 \leq i \leq p$. A common supersequence of R is a string $\hat{S}$ such that $\hat{S}>S_i$ for each i, $1 \leq i \leq p$.

We shall often refer to common subsequences and common supersequences as common sequences.

Definition 3. A longest common subsequence of R is a common subsequence $\tilde{S}$ such that no other common subsequences are longer than $\tilde{S}$. Similarly, a shortest common supersequence $\hat{S}$ such that no other common supersequences are shorter than $\hat{S}$.

The following notations will be used throughout the paper.

Notation.

$\widetilde{CS}(R)$ A set of common subsequences of R.

LCS(R) A set of longest common subsequences of R.

$\widehat{CS}(R)$ A set of common supersequences of R.

SCS(R) A set of shortest common supersequences of R.

Definition 4. A string S is primitive if no character appears more than once in S.

Example 1. Let Σ={a, b, c, d, e} be an alphabet. Let R={abcde, acbed, badce} be a set of 3 strings over Σ. Note that each string of R is primitive. Then,

$\widetilde{CS}(R)$={ace, ac, ad, ae,...},

LCS(R)={ace},

$\widehat{CS}(R)$={babdcbede, abadcbede, abcadebced,...},

and

SCS(R)={babdcbede, abadcbede,...}.

Since the problems of finding a shortest common supersequence and a longest common subsequence of the set R include the term "shortest" and "longest", we can consider both problems as optimization problems. In measuring the computational complexity of the optimization problem it is sufficient to discuss the one of the corresponding decision problems. Now we define a yes/no version of the problem of finding common sequences of of the set R as follows [7,14].

Definition 5. The yes/no LCS problem is defined as follows: Given a positive integer k and a set $R=\{S_1,S_2,...,S_p\}$ $(p\geq 2)$ of strings over an alphabet Σ, is there a string $\tilde{S} \in \widetilde{CS}(R)$ such that $|\tilde{S}|\geq k$?

Definition 6. The yes/no SCS problem is defined as follows: Given a positive integer k and a set R of strings over an alphabet Σ, is there a string $\hat{S} \in \widehat{CS}(R)$ such that $|\hat{S}|\leq k$?

3. LCS Problems

In this section we consider the computational complexity of the following decision problem.

Definition 7. The yes/no primitive p-LCS problem (shortly, the primitive LCS problem) is defined as follows: Given an integer k and a set $R=\{S_1,S_2,...,S_p\}$ $(p\geq 2)$ of primitive strings over an alphabet Σ, is there a string $\tilde{S} \in \widetilde{CS}(R)$ such that $|\tilde{S}|\geq k$?

This problem is an extension of Szymanski's primitive 2-LCS problem [12]. Define $N=\sum_{i=1}^{p}|S_i|$. Then we get the next theorem.

Theorem 1. The primitive LCS problem is solved in $O(N^3)$ time.

We prove Theorem 1 by presenting an algorithm that finds a longest common subsequence of R in $O(N^3)$ time [6,14].

4. SCS Problems

In what follows, we define two restricted SCS problems: the one with the restricted length of the strings in R, and the other with the restricted occurrence of the character in each string in R.

<u>Definition 8.</u> The length restricted yes/no SCS problem (shortly, the problem S1) is defined as follows: Given a positive integer k and a set $R=\{S_1,S_2,\ldots,S_p\}$ $(p\geq 2)$ of strings such that $|S_i|=5$ for each i, $1\leq i\leq p-1$, is there a string $\hat{S}\in \hat{CS}(R)$ such that $|\hat{S}|\leq k$?

<u>Definition 9.</u> The primitive yes/no SCS problem (shortly, the problem S2) is defined as follows: Given a positive integer k and a set $R=\{S_1,S_2,\ldots,S_p\}$ $(p\geq 2)$ of primitive strings such that $|S_i|=5$ for each i, $1\leq i\leq p-2$, is there a string $\hat{S}\in \hat{CS}(R)$ such that $|\hat{S}|\leq k$?

We get the following two theorems. (The proofs of theorems are given in [5,14].)

<u>Theorem 2.</u> The problem S1 is NP-complete.

<u>Theorem 3.</u> The problem S2 is NP-complete.

These results tell us that it is hard to develop an efficient algorithm for the SCS problem. The problem of finding a shortest common supersequence is of course in NP-hard optimization problem. Generally, if we are to produce an algorithm of low polynomial complexity to solve an NP-hard optimization problem, then it will be necessary to relax the meaning of the solution. One of these relaxations is to allow an approximate solution.

Let $R=\{S_1,S_2,...,S_p\}$ be a set of strings. We assume without loss of generality that $|S_i|=n$ for each i, $1\leq i\leq p$. Let $SCS(S_1,S_2,...,S_k)$ be a function which returns a single representative of the set SCS(R') where $R'=\{S_1,S_2,...,S_k\}$. By $|SCS(S_1, S_2,...,S_k)|$ we mean the length of a $SCS(S_1,S_2,...,S_k)$.

We consider the following approximation algorithm, called APPRO1, which is described as the ALGOL-like program. The input of this algorithm is the set $R=\{S_i|1\leq i\leq p, |S_i|=n\}$ of strings and the output is a string T_1.

```
Procedure APPRO1
begin
  $T_1:=S_1$;
  for i:=2 to p do $T_1:=SCS(T_1,S_i)$
end.
```

It is clear that the output of APPRO1 is a common supersequence of R. The function $SCS(T_i,T_j)$ can be easily implemented by using the algorithm given by Wagner and Fischer [13]. This computation of $SCS(T_i,T_j)$ takes at most $O(n^2)$ time. Thus the total time of APPRO1 is $O(n^2p)$.

We can get the following theorem on APPRO1 [14].

Theorem 4. Let $R=\{S_i | 1\leq i\leq p, |S_i|=n\}$ $(p\geq 2)$ be a set of strings. Let $n+m$ be the length of T_1 defined by APPRO1. Let $n+m^*$ be the length of the shortest common supersequence of R. Then,

$$m\leq (p-1)\cdot m^*.$$

Now, we consider the correspondence between the LCS and SCS problems, especially the sets LCS(R) and SCS(R). Maier [7] presented the following question in his paper.

Question [Maier]. Do there exist low order polynomial reductions directly between the LCS and SCS problems?

We divide the discussion into two cases, $p=2$ and $p>2$, where p is the cardinality of the set R. First, we discuss the case of $p=2$. In this case we can get the following lemma [14].

Lemma 1. Let $R=\{S_1,S_2\}$ be a set of two strings such that $|S_1|=m$ and $|S_2|=n$. Let $l=|LCS(R)|$ and $k=|SCS(R)|$, respectively. Then,

$$k=m+n-l.$$

By Lemma 1, we can get the answer of the LCS problem immediately when we get the answer of the SCS problem and vice versa.

Theorem 5. Let R be a set of two strings. Then, there exists a polynomial reduction betweeen the LCS and SCS problems.

Next, we discuss the case of $p \geq 2$. We consider the primitive strings rather than the general strings. Let $R=\{S_1, S_2, \ldots, S_p\}$ $(p \geq 2)$ be a set of strings such that each string S_i $(1 \leq i \leq p)$ is primitive. In Section 3, we have shown that the primitive LCS problem is polynomially solvable. On the other hand, we have shown that the primitive SCS problem is NP-complete. Note that in the definition of the primitive SCS problem (Definition 9), we restricted the length of strings. But we can easily prove that the primitive SCS problem, in which there is no restriction on the length of each string, is also NP-complete.

Now, assume that there exists a polynomial reduction between the LCS and SCS problems. Then, we can construct a polynomial time algorithm for the primitive SCS problem, which at first reduces the SCS problem into the LCS problem and then computes the answer. But we have shown that the primitive SCS problem is NP-complete. Thus, in general case, we get the following conjecture for Maier's open problem.

Conjecture. There don't exist polynomial reductions between the LCS and SCS problems.

5. Application of Common Supersequences

In this section we discuss the placement and routing at the design of PWB's [11] as one of the applications of common supersequences. We have already shown that the problem of finding a shortest common supersequence of an arbitrary set of strings is intractable. So, we must develop a good approximation algorithm in order to use the shortest common supersequences in the practical applications. Firstly, we introduce the new mathematical model, called ARG, for the problem of finding a shortest common supersequences.

Maier gave the threading scheme [7] as model for the problem of finding common supersequences. This model is useful to prove theoretical results. In this section, we introduce a new mathematical model, called ARG, defined as follows.

Definition 10. Let $R=\{S_1,S_2,\ldots,S_p\}$ be a set of strings over an alphabet Σ and

$N=\{(i,j)\,|\,S_i=s_{i_1}s_{i_2}\ldots s_{i_j}\ldots s_{i_{|S_i|}},\ s_{i_j}\in\Sigma,\ 1\le i\le p,\ 1\le j\le |S_i|\}$.

An Acyclic Representation Graph (ARG) for a set R is an acyclic directed graph $G=(V,E)$ such that

(1) $V=\{v\}$ where $|V|\le|N|$ and there exists a mapping h from N onto V such that (a) $h(i,j)\ne h(i,j')$, and (b) $h(i,j)=v=h(k,l)$, $i\ne k$, for some $v\in V$ only if $s_{i_j}=s_{k_l}$.

(2) $E=\{(v,v')\,|\,v=h(i,j),\ v'=h(i,j+1),\ 1\le i\le p,\ 1\le j\le |S_i|-1\}$.

Example 2. Let $R=\{S_1,S_2,S_3\}$ be a set of three strings such that S_1=abcd, S_2=acbd and S_3=adbc. Then, $N=\{(i,j)\,|\,1\leq i\leq 3, 1\leq j\leq 4\}$. Figure 1 shows an ARG $G=(V,E)$ of R. The mapping $h:N\to V$ is defined as follows:

$(1,1)\to v_1$, $(1,2)\to v_3$, $(1,3)\to v_4$, $(1,4)\to v_6$

$(2,1)\to v_1$, $(2,2)\to v_4$, $(2,3)\to v_5$, $(2,2)\to v_6$

$(3,1)\to v_1$, $(3,2)\to v_2$, $(3,3)\to v_3$, $(3,4)\to v_4$.

Note that, under the mapping h, V and E are assumed to be specified as follows:

$V=\{v_i\,|\,1\leq i\leq 6\}$,

$E=\{(v_1,v_3),\ (v_3,v_4),\ (v_4,v_6),\ldots,(v_2,v_3)\}$.

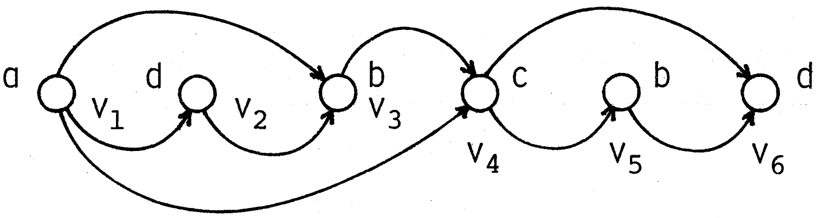

Fig. 1 ARG G=(V,E).

Before explaining the correspondence between ARG and the common supersequences, we show the property that ARG has.

Property. Let R be a set of strings and $G=(V,E)$ be an ARG of R. Let $\hat{S}$ be a string which is obtained by applying the topological sorting [10] to G. Then, $\hat{S}$ is a common supersequence of R such that $|\hat{S}|=|V|$.

Definition 11. The yes/no ARG problem is defined as follows: Given a set R of strings and a positive integer k, is there an ARG G=(V,E) of R such that $|V| \leq k$?

We get the following theorem on the yes/no ARG problem.

Theorem 6. The yes/no ARG problem is NP-complete.

Theorem 6 indicates that the problem of finding a minimum ARG is intractable. But for the primitive strings, we can find a minumum ARG in polynomial time. Once we get the minimum ARG, we can easily find the shortest common supersequence of R by applying the topological sorting to G.

Theorem 7. Let R be a set of primitive strings over an alphabet Σ. Let ρ be a relation on $\Sigma \times \Sigma$ which is defined by $(a,b) \in \rho$ if and only if "ab" is a substring of some primitive string in R. If the relation ρ is a partial ordering, then a shortest common supersequence of R is found in polynomial time.

Now we give an approximation algorithm based on ARG. The following is an outline of our algorithm SCSARG.

```
Algorithm SCSARG (Outline)
begin
  initialize;  / construct an ARG of $S_1$ /
  for i:=2 to p do
  begin
```

```
    match;  / determine the maximum matching between the ARG
              of $\{S_1,S_2,\ldots,S_{i-1}\}$ and $S_i$ /
    constructarg  / construct an ARG of $\{S_1,S_2,\ldots,S_i\}$ /
  end;
  topologicalsort  / find a common supersequence of
                     $\{S_1,S_2,\ldots,S_p\}$ /
end.
```

The maximum matching is defined as follows. Let G be an ARG of the set R of strings over an alphabet Σ. Let S be a string over the alphabet Σ. The maximum matching between G and S is a longest subsequence S' of S such that we can construct the ARG G' of $R \cup \{S'\}$ without adding any vertices to G.

Next we discuss the distinguished features of our algorithm SCSARG. Let $R=\{S_1,S_2,\ldots,S_p\}$ and $N=\sum_{i=1}^{p} |S_i|$, as before. The merits of algorithm SCSARG are summarized as follows:

(i) The result of SCSARG doesn't include redundancy. In other words, if we determine the string S as a common supersequence of R, there is no other string S' such that $S'<S$ and S' is also a common supersequence.

(ii) For $p=2$, SCSARG produces the exact, i.e. optimal solution.

(iii) For R which satisfies the condition of Theorem 7, SCSARG produces the exact solution, also.

Note that (i) and (iii) are not true in the algorithm APPRO1 given in Section 4. Concerning the evaluation of algorithm, we show the following theorem.

Theorem 8. Let $R=\{S_1,S_2,...,S_p\}$ be a set of strings such that $|S_i|=n$ $(1\leq i\leq p)$. Let $n+m$ be the length of a common supersequence of R, that is determined by SCSARG. Let $n+m^*$ be the length of the shortest common supersequence of R. Then,

$$m\leq(p-1)\cdot m^*.$$

Now we propose a new method to solve the placement and routing problem for two-sided Printed Wiring Boards (PWB's). Our method is based on the concept of the shortest common supersequences discussed in this paper. The placement and routing problem is one of the most important problems in the design of electronic systems fabricated on-board and many investigations have already been done. Here we propose a new method for this problem. Figure 2 shows the general flow chart of our method.

The characteristics of our method is summarized as follows:

(1) Both placement and routing are determined almost simultaneously. (Placement is determined before routing in conventional methods.)

(2) We never fail to determine routing if there is no physical restriction such as the area of the board. That is, we get 100 percent routing.

(3) But, there are some more redundant lines in their length in the result obtained than those in conventional methods.

Most of the conventional methods contain the heuristic techniques such as the maze-running method. By now there is few algorithmic approach to the PWB problem. So, we think our method is noteworthy investigation since it is algorithmic.

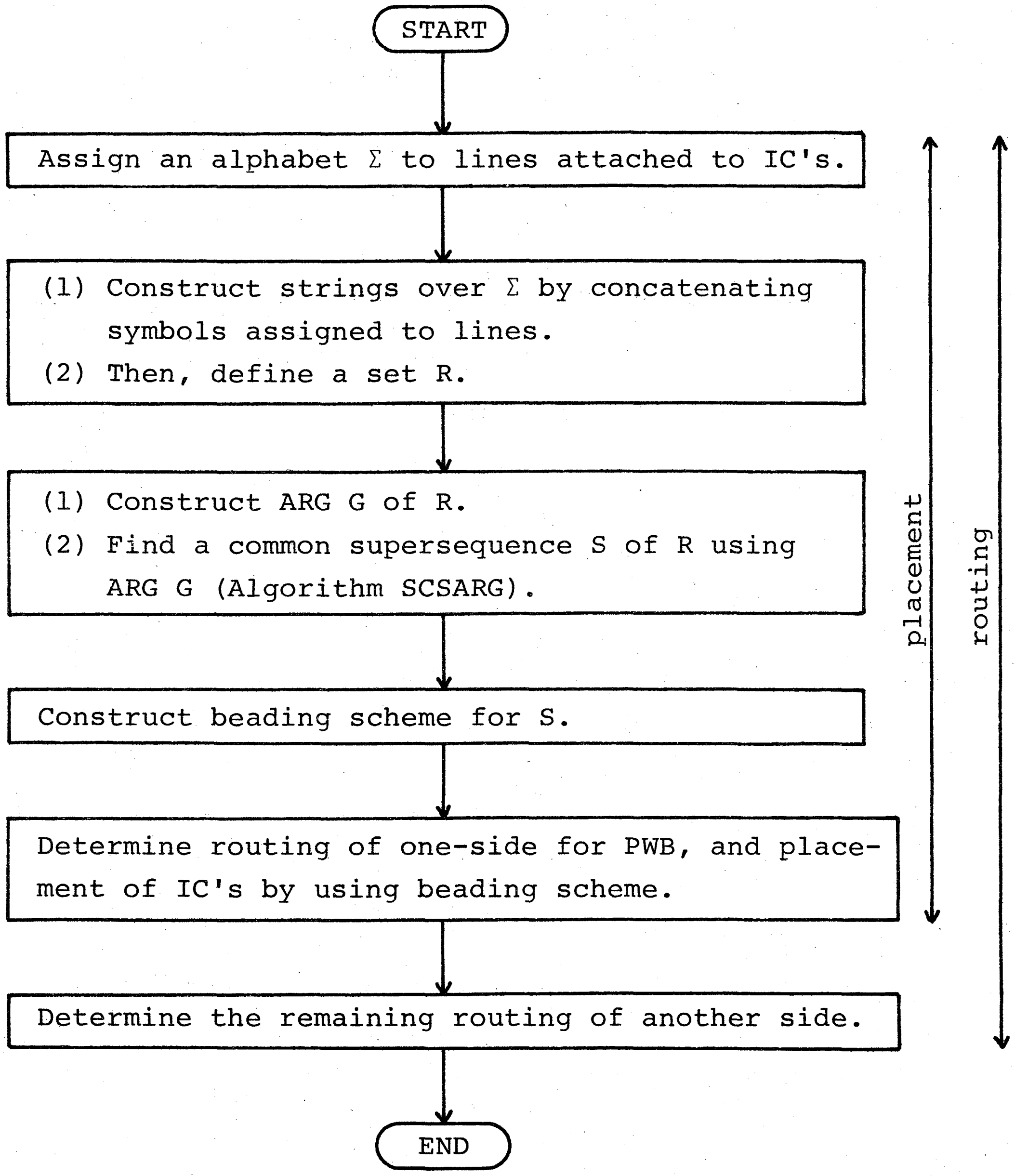
START
Assign an alphabet Σ to lines attached to IC's.
(1) Construct strings over Σ by concatenating symbols assigned to lines.
(2) Then, define a set R.
(1) Construct ARG G of R.
(2) Find a common supersequence S of R using ARG G (Algorithm SCSARG).
Construct beading scheme for S.
Determine routing of one-side for PWB, and placement of IC's by using beading scheme.
Determine the remaining routing of another side.
END
placement
routing

Fig. 2 Flow chart.

6. Conclusion

The computatinal complexity of the LCS and SCS problems is mainly discussed in this paper. In Section 3, we present a polynomial time algorithm to solve the primitive LCS problem. However, in Section 4, we show that two restricted versions of the yes/no SCS problems still remain to be NP-complete. The results obtained in this paper, including that given in [7,9] are summarized in Table 1.

In Section 5, an application of the SCS problem (strictly, common supersequences) is considered. We define a new model ARG to solve the problem of finding a shortest common supersequence. An effective approximation algorithm using ARG model is presented. Then, we try to make a new approach, which is based on common supersequences, to the design of the PWB's.

Now we survey briefly the directions of future researches related. First, on the SCS problems, it is expected to make clear the boundary between P and NP (that is a class of problems in P and a class of problems in NP).

Secondly, in order to apply common sequences to the practical applications, it is desirable to develop more effective approximation algorithm to find common sequences than that ever proposed.

Thirdly, much more applications of common sequences should be considered.

Table 1 Computational complexity of decision problems.

Problem	LCS problem	SCS problem
(general)	NP-complete [7]	NP-complete [7]
$\|\Sigma\|$=const.	NP-comlete for $\|\Sigma\|=2$ [7]	NP-complete for $\|\Sigma\|=2$ [9]
$\|S_i\|$=const.	Polynomial time (by definition)	NP-complete for $\|S_i\|=5$ $(1\le i\le p-1)$
primitive	Polynomial time	NP-complete for $\|S_i\|=5$ $(1\le i\le p-2)$

References

[1] Aho,A.V., Hopcroft,J.E. and Ullman,J.D.:"The Design and Analysis of Computer Algorithms", Addison-Wesley, Reading, Mass. (1974).

[2] Hirschberg,D.S.:"Algorithms for the longest common subsequence problem", J.Assoc.Comput.Mach., 24, 4, pp.664-675 (1977).

[3] Hirschberg,D.S.:"Complexity of common subsequence problems", Lecture notes in computer science, 56 (1977).

[4] Horowitz,E. and Sahni,S.:"Fundamentals of Computer Algorithms", Computer Science Press, Maryland (1978).

[5] Kikuno,T., Yoshida,N. and Wakabayashi,S.:"The complexity of some decision problems on common supersequences", Trans. of IECEJ, to appear (J64-D, 1981) in Japanese.

[6] Kikuno,T., Yoshida,N. and Wakabayashi,S.:"Computational complexity of longest common subsequence problem", to be submitted to IEEE Trans. on PAMI.

[7] Maier,D.:"The complexity of some problems on subsequences and supersequences", J.Assoc.Comput.Mach., 25, 2, pp.322-336 (1978).

[8] Needleman,S.B. and Wunsch,C.D.:"A general method applicable to the search for similarities in the amino acid sequence of two proteins", J.Molecular Biol., 48, pp.443-453 (1970).

[9] Räihä,K.-J. and Ukkonen,E.:"The shortest common supersequence problem over binary alphabet is NP-complete", Report C-1979-95, Dept. of Computer Science, Univ. of Helsinki (1979).

[10] Reingold,E.M., Nievergelt,J. and Deo,N.:"Combinatorial Algorithms: Theory and Practice", Prentice-Hall, New Jersey (1977).

[11] Research committee on the design technology of electronic equipments:"Computer aided design of electronic equipments (3)-The recent trends of the CAD technology in physical design-", Proc. IPSJ, 21, 1, pp.50-61 (1980) in Japanese.

[12] Szymanski,T.G.:"A special case of the maximal common subsequence problem", TR-170, Comptr. Sci. Lab., Princeton U., Princeton, N.J. (1975).

[13] Wagner,R.A. and Fischer,M.J.:"The string-to-string correction problem", J.Assoc.Comput.Mach., 21, 1, pp.168-173 (1974).

[14] Wakabayashi,S.:"Computatinal complexity on common sequence problem", Thesis for MS in Electrical Systems Eng., Hiroshima University (1981).