On the Approximation of Finding Various Minimal, Maximal, and Consistent Sequences

Martin Middendorf

Institut für Angewandte Informatik und Formale Beschreibungsverfahren,
Universität Karlsruhe, D-76128 Karlsruhe, Germany
e-mail: mmi@aifb.uni-karlsruhe.de

Abstract. In this paper we investigate the complexity of finding various kinds of common super- and subsequences with respect to one or two given sets of strings. We show how these problems can be related to finding sequences having a fixed character composition. This leads to a unified approach for characterizing the complexity of such problems. Moreover, we derive interesting results on the approximability of problems which are dual to the well known Shortest Common Supersequence and Longest Common Subsequence problems.

1 Introduction

In this paper we investigate the complexity of finding various kinds of common super- and subsequences with respect to one or two given sets of strings. Problems for supersequences and subsequences find applications in different areas, e.g. mechanical engineering and molecular biology. Recently, there has been a growing interest to study such problems.

A supersequence of a string S is any string that is obtained by inserting characters into S; a subsequence of S is any string obtained by deleting characters from S. A non-supersequence (non-subsequence) of S is a string that is not a supersequence (subsequence) of S. A (common) supersequence of a set L of strings is a supersequence of every string in L. (Common) subsequences, non-supersequences, and non-subsequences of a set of strings are defined similarly. A supersequence is minimal if none of its proper subsequences is still a supersequence. A subsequence is maximal if none of its proper supersequences is still a subsequence. Maximal non-supersequences and minimal non-subsequences are defined similarly.

It is known that the decision versions of the problems Shortest Common Supersequence (SCS), Longest Common Subsequence (LCS), Longest Common Non-Supersequence (LCNS), and Shortest Common Non-Subsequence (SCNS) are NP-complete even over a binary alphabet [3,4,7,8,10]. It is also known that all these problems are MAX SNP-hard over an alphabet of arbitrary size [10]. Thus, it is likely that there exists no polynomial time approximation scheme for them. It is an interesting open problem whether these problems remain MAX SNP-hard when the size of the alphabet is a constant. SCS, LCS, LCNS, and SCNS deal with finding a sequence consistent with respect to only one given set

of strings. Problems of finding a sequence consistent with respect to two given sets of strings have also been studied in [2,4,10].

In this paper we show how the problem of finding sequences consistent with two given sets of strings can be related to the problem of finding sequences that have a fixed character composition and that are consistent with only one given set of strings. This leads to a unified approach allowing to characterize the complexity of such problems in a much clearer way than has been done before. Our results complete those of [2,3,4,7,8,10] and disprove a conjecture of [2]. Moreover, we derive interesting approximability results on problems which are dual to SCS, LCS, and LCNS.

The first part of this paper is devoted to the problem of finding minimal supersequences, maximal subsequences, maximal non-supersequences, and minimal non-subsequences. A simple strategy for finding a minimal supersequence for a set L of strings is to start with any supersequence S of L (e.g. with the trivial supersequence which consists of the concatenated strings in L) and then to shorten this supersequence. This shortening can be done by iteratively trying to erase symbols from S until no more symbols of S can be erased without violating the property that S is a supersequence of L. To sum up, finding any minimal supersequence can be done in polynomial time, but even to determine the length of a shortest one is NP-complete. Thus, measuring the quality of any minimal supersequence as an approximation for a shortest one cannot be done by comparing its length to the length of a shortest one. But, what about comparing the length of any supersequence against the length of the longest minimal supersequence? How hard is it to find a longest minimal supersequence? Can longest minimal supersequences be approximated? We show that the problems of finding longest minimal supersequences, shortest maximal subsequences, and shortest maximal non-supersequences are MAX SNP-hard even over a binary alphabet. Remember, that it is not known whether SCS, LCS, and LCNS are MAX SNP-hard if the size of the alphabet is a constant. We leave open whether finding longest minimal non-subsequences is MAX SNP-hard over binary alphabet.

In the second part we study the problem of finding common (non-)supersequences and (non-)subsequences which have a character composition that is (partially) fixed by the instance. Not surprisingly, over a binary alphabet we show that finding such sequences with a fixed number of ones and zeros is NP-complete. If only the number of zeros is fixed we show that several corresponding optimization problems are MAX SNP-hard.

The third part of the paper deals with the problem of finding sequences with respect to two given sets of strings. Several authors have studied such problems (comp. [2,4,10]). Middendorf [4] examins the problem of finding for two given sets of strings a sequence that is a subsequence of one set and a non-subsequence of the other set. Jiang and Li [2] investigated th following problem in Valiant's pac-learning model (See [6,9] for an exact definition of the pac-learning model): *Learning a supersequence*. For a target sequence S a positive example is a subsequence of S, and a negative example is a sequence which is not a subsequence

of S. Given a set of positive and negative examples drawn according to a fixed distribution, the aim is to find a sequence that classifies future positive and negative examples (from the same distribution) approximately correctly.

Jiang and Li [2] showed that sequences cannot be learned by sequences in the distribution-free pac-learning model of Valiant, assuming RP$\neq$NP. Applications for the problem are given in [2].

Actually, Jiang and Li proved that given positive and negative examples, it is an NP-complete problem even to find any sequence that is a supersequence of each sequence in a given set POS of positive examples and that is not a supersequence of any sequence in a given set NEG of negative examples. They showed that the problem remains NP-complete if there are only two positive examples (Zhang [10] showed that finding a longest such sequence if only one positive example is given is MAX SNP-hard). On the other hand, they found a polynomial time algorithm for the problem, if there is only one negative example. They conjectured that the problem is polynomial time solvable for any constant number of negative examples. Here we disprove this conjecture (unless P=NP) by showing that the problem is NP-complete.

We define such problems in a more general setting: Given a set of strings L over an alphabet Σ. A sequence S over Σ is of type $Super$ (resp. Sub, $NSuper$, $NSub$) with respect to L if S is a supersequence (resp. subsequence, non-supersequence, non-subsequence) of L. Given a pair $\mathcal{L} = (L_1, L_2)$ of sets of strings over Σ a sequence S over Σ is of type (x_1, x_2), $x_i \in \{Super, Sub, NSuper, NSub\}$ with respect to $\mathcal{L}$ if S if a sequence of type x_i with respect to L_i. The Consistent Supersequence problem (CCS) is to find, given a pair $\mathcal{L}$ of sets of strings, a sequence that is of a given type with respect to $\mathcal{L}$. We investigate the complexity of the CCS problem for all different types of sequences and with respect to the number of strings in L_2.

2 Minimal and Maximal Sequences

For an integer k, $[1:k]$ denotes the set of integers between 1 and k. Let a string $S = s_1 s_2 \ldots s_{|S|}$ be a subsequence of a string $T = t_1 t_2 \ldots t_{|T|}$. An *embedding* of S in T is a strong growing function f from $[1:|S|]$ to $[1:|T|]$ such that $s_i = t_{f(i)}$ for all $i \in [1:|S|]$. We say that s_i is *mapped onto* $t_{f(i)}$ by f, $i \in [1:|S|]$. An embedding is *leftmost* if, for every embedding g of S in T, we have $f(i) \leq g(i)$ for all $i \in [1:|S|]$

In this section we show that several optimization problems concerning minimal and maximal sequences are MAX SNP-hard even over a binary alphabet. The class MAX SNP was introduced by Papadimitriou and Yannakakis [5]. Every problem in this class can be approximated with a constant factor. There are hard problems in this class with respect to $L-reductions$: For a polynomial time transformation f from an optimization problem Π to an optimization problem Π' the transformation f is called L-reduction (linear reduction) if there are constants α, β such that:

i) For an instance P of Π we have $opt(f(P)) \leq \alpha \cdot opt(P)$ where $opt(P)$ is the cost of the optimal solution for P.
ii) For any solution of $f(P)$ with cost c a solution of P with cost c' can be found in polynomial time such that $|c' - opt(p)| \leq \beta |c - opt(f(P)|$.

L-reductions preserve approximability in the following sense: if Π can be L-reduced to Π' and there is a polynomial time approximation for Π' with error ϵ then there is also one for Π with error $\alpha\epsilon\beta$. Hence, if there is a polynomial time approximation scheme (PTAS) for Π', then so for Π. A problem is *hard* for MAX SNP if every problem in MAX SNP can be L-reduced to it. Therefore, if a problem is MAX SNP-hard and there is a PTAS for Π then so for all problems in MAX SNP. It is quite unlikely that a MAX SNP-hard problem has a PTAS because this would imply P=NP (see [1]).

Theorem 1 *The following problems are MAX SNP-hard over a binary alphabet:*

a) Longest Minimal Common Supersequence.
b) Shortest Maximal Common Subsequence.
c) Shortest Maximal Common Non-Supersequence.

Proof. We L-reduce the Dominating and Independent Set-B problem to each of our problems a), b), and c) (The proof of b) is omitted). It is not hard to show that Dominating and Independent Set-B is MAX SNP-hard by an L-reduction from the Dominating Set-B problem which is known to be MAX SNP-hard [5]. Let a graph $G = (V, E)$ of bounded degree B with node set $V = \{v_1, v_2, \ldots, v_n\}$ and edge set $E = \{e_1, e_2, \ldots, e_m\}$ be given. Clearly, any smallest dominating and independent set of G is of size at least $\frac{n}{B+1}$.

a) We construct the following set of strings over the alphabet $\{0, 1\}$: Define

$$S_0 = (10)^{2n+1}.$$

For every edge $e_l = \{v_i, v_j\} \in E$, $l \in [1 : m]$, $i < j$ define

$$T_l = (1010)^{i-1}10110(1010)^{j-i-1}11010(1010)^{n-j}.$$

Set $L = \{S_0\} \cup \{T_1, T_2, \ldots, T_m\}$. Observe, that each string T_l contains exactly $2n$ zeros and $2n + 2$ ones. From the construction follows that no minimal supersequence of L contains the substring 00 or the substring 111. Furthermore, every minimal supersequence of L has a one as its leftmost symbol and a zero as its rightmost symbol. It follows that every minimal supersequence of L that contains $\geq 2n+2$ zeros has $(10)^{2n+2}$ as a subsequence. Since every string in L is embeddable in the string $(10)^{2n+2}$, it is the only string with $\geq 2n+2$ zeros that is a minimal supersequence of L. On the other hand, every supersequence of S_0 contains at least $2n + 1$ zeros. We derive that every minimal supersequence of L containing $2n + 1$ zeros is of the form $(*)$ $x_1 0 x_2 0 \ldots 0 x_{2n+1} 0$ where $x_i = 1$ or $x_i = 11$ for $i \in [1 : 2n + 1]$.

Claim 1 *The string T_l with $e_l = \{v_i, v_j\}$, $i < j$ is embeddable in a string of the form $(*)$ iff $x_{2i} = 11$ or $x_{2j} = 11$.*

Proof. Assume $x_{2i} = 1$ and $x_{2j} = 1$. Consider a leftmost embedding of T_l in a string of the form $(*)$. The $2j+1$th one is mapped onto a one in x_{2j+1}. In T_l there are $2(n-j)+2$ zeros to the right of the $2j+1$th one whereas in the string of the form $(*)$ there are only $2(n-j)+1$ zeros to the right of x_{2j+1}. This is a contradiction. The other direction of the proof is obvious. □

Using Claim 1 we get:

Claim 2 *A supersequence of L of the form $(*)$ is minimal iff for all $j \in [1:n+1]$ we have $x_{2j-1} = 1$ and for all $i \in [1:n]$ with $x_{2i} = 11$ there is a string $T_l \in S$ with $e_l = \{v_i, v_j\}$, $i \neq j$ such that $x_{2j} = 1$.*

In the following we show that there is a dominating and independent set $V' \subset V$ of size k for G iff there is a minimal supersequence of L of length $5n+2-k$.

Let V' be a dominating and independent set for G, and S be a string of the form $(*)$ with $x_j = 1$ for $j \in [1:2n+1]$ with the exception of those $j = 2i$ with $v_i \in V - V'$ for which $x_{2i} = 11$, $i \in [1:n]$. Since V' is an independent set and by Claim 3 it follows that S is a supersequence of L. Then, since V' is a dominating set and by Claim 2 it follows that S is a minimal supersequence of L of length $\geq 5n+2-k$.

W.l.o.g. assume $n-k > 2$. Then, a minimal supersequence of L of length $5n+2-k$ is of the form $(*)$. By Claim 2 we have $x_{2i+1} = 1$ for $i \in [0:n]$. Also, there are $i_1, i_2, \ldots, i_{n-k}$ with $x_{2i_j} = 11$ for $j \in [1:n-k]$. Set $V' = V - \{v_{i_1}, v_{i_2}, \ldots, v_{i_{n-k}}\}$. Now, Claim 1 implies that V' is an independent set and Claim 2 implies that V' is a dominating set for G.

Altogether, the optimal solution for L has length $\leq 5n+2-opt(G) \leq (5B+7)opt(G)$, where $opt(G)$ is the size of the optimal solution for G. Thus, we have an L-reduction with $\alpha = 5B+7$ and $\beta = 1$.

c) We construct a set L of strings over $\{0,1\}$ as follows: Define

$$S_0 = (1^{2n}0)^{n-2}1^{2n}, \qquad S_i = 0^{i-1}1^20^{n-i} \text{ for } i \in [1:n], \qquad S_{n+1} = 0^n.$$

For every edge $e_l = \{v_i, v_j\} \in E$, $i < j$, $l \in [1:m]$ define

$$T_l = 0^{i-1}10^{j-i}10^{n-j}.$$

Set $L = \{S_i \mid i \in [0:n+1]\} \cup \{T_1, T_2, \ldots, T_m\}$. Clearly, every non-supersequence of S_{n+1} contains at most $n-1$ zeros. A non-supersequence of L that contains $\leq n-2$ zeros is maximal only if it contains at least $2n$ ones and thus has length $\geq 2n$. A non-supersequence of $\{S_i \mid i \in [1:n]\}$ containing exactly $n-1$ zeros cannot contain the substring 11. Now, it is not difficult to see that G has a dominating and independent set $V' = \{v_{i_1}, v_{i_2}, \ldots, v_{i_k}\}$, $i_1 < i_2 < \ldots < i_k$

iff $0^{i_1-1}10^{i_2-i_1}1\ldots10^{n-i_k}$ is a maximal non-supersequence of L. Hence, the optimal solution for L has length $\leq n-1+opt(G) \leq (B+2)opt(G)$, where $opt(G)$ is the size of the optimal solution for G. Thus, we have an L-reduction with $\alpha = B+2$ and $\beta = 1$. □

It is left open whether finding a longest minimal non-subsequence is MAX SNP-hard over a binary alphabet.

3 Sequences with a Fixed Character Composition

In this section we address optimization problems of finding (non-)supersequences and (non-)subsequences which have a character composition that is (partially) fixed by the instance. Such problems are not only of theoretical interest. They have applications in molecular biology and other fields. For an example, in order to determine the sequence of the components of a macromolecule from subsequence information it may sometimes be possible to determine also the total number of occurences of each component in the molecule (e.g. the number of adenine, cytosine, guanine, and thymine bases in a DNA sequence). First we consider optimization problems over the alphabet $\{0,1\}$ where only the number of zeros is fixed by the instance. Trivially, for larger alphabets similar results hold.

Theorem 2 *The following problems are MAX SNP-hard over a binary alphabet if the number of zeros is fixed by the instance:*

a) Shortest Common Supersequence,
b) Longest Common Subsequence,
c) Longest Common Non-Supersequence,
d) Shortest Common Non-Subsequence,
e) Longest Minimal Common Non-Subsequence.

Proof. a) We L-reduce the Vertex Cover-B problem to our problem. Let a graph $G=(V,E)$ of bounded degree B with node set $V=\{v_1,v_2,\ldots,v_n\}$ and edge set $E=\{e_1,e_2,\ldots,e_m\}$ be given. Clearly, any smallest vertex cover of G is of size at least $\frac{n}{B+1}$. We construct a set L of strings over $\{0,1\}$ as follows: Define

$$S_0 = 1(001)^n.$$

For every edge $e_l=\{v_i,v_j\} \in E$, $i<j$, $l \in [1:m]$ define

$$T_l = 0^{2i-1}10^{2(j-i)-1}10^{2(n-j)+1}.$$

Set $L=\{S_0\}\cup\{T_1,T_2,\ldots,T_m\}$. Observe, that T_l, $l\in[1:m]$ contains exactly $2n-1$ zeros and S_0 contains exactly $2n$ zeros. Now, it is not difficult to see that G has a vertex cover $V'=\{v_{i_1},v_{i_2},\ldots,v_{i_k}\}$, $i_1<i_2<\ldots<i_k$ iff $1(001)^{i_1-1}0101(001)^{i_2-i_1-1}0101\ldots0101(001)^{n-i_k}$ is a supersequence of L of length $3n+1+k$ containing exactly $2n$ zeros. Hence, the optimal solution for L

has length $\leq 3n + 1 + opt(G) \leq (3B + 5)opt(G)$, where $opt(G)$ is the size of the optimal solution for G. Thus, we have an L-reduction with $\alpha = 3B+5$ and $\beta = 1$.

b), and c) can be shown by L-reducing the Independent Set-B problem and d) can be shown by L-reducing the Vertex Cover-B problem. The corresponding proofs are omitted.

e) We L-reduce the Dominating Set-B problem to our problem. Given an instance $G = (V, E)$ of Dominating Set-B we we construct a set L of strings over $\{0, 1\}$. Clearly, any smallest dominating set of G is of size at least $\frac{n}{B+1}$. Define

$$S_0 = (1^2 0)^{n-2} 1^2.$$

For every edge $e_l = \{v_i, v_j\} \in E$, $i < j$, $l \in [1 : m]$ define

$$T_l = (10)^{i-1} 0 (10)^{j-i-1} (01)^{n-j}$$

If a minimal non-subsequence S of L with exactly $n - 1$ ones contains 11 as substring, then it must be of the form $0^{i-1}1^2 0^{n-i}$ for an $i \in [1 : n]$. Thus it has length $n + 1$. It is not difficult to see that G has a dominating set $V' = \{v_{i_1}, v_{i_2}, \ldots, v_{i_k}\}$, $i_1 < i_2 < \ldots < i_k$ iff $0^{j_1-1}10^{j_2-j_1}1 \ldots 10^{n-j_{n-k}}$ is a minimal non-subsequence of L containing exactly $n - 1$ zeros that has length $2n - 1 - k$ where $\{j_1, j_2, \ldots, j_{n-k}\} = \{1, 2, \ldots, n\} \cap \{i_1, i_2, \ldots, i_k\}$, $j_1 < j_2 < \ldots < j_{n-k}$. Hence, the optimal solution for L has length $\leq 2n - 1 - opt(G) \leq 2Bopt(G)$, where $opt(G)$ is the size of the optimal solution for G. Thus, we have an L-reduction with $\alpha = B$ and $\beta = 1$. □

As a corollary we easily derive:

Corollary 1 *Given a set L of strings over a binary alphabet and integers n, m. The following problems are NP-complete:*

a) Find a supersequence of L containing exactly n zeros and m ones.
b) Find a subsequence of L containing exactly n zeros and m ones.
c) Find a non-supersequence of L containing exactly n zeros and m ones.
d) Find a non-subsequence of L containing exactly n zeros and m ones.

Note, that we have given with this corollary quite simple proofs for the known facts that SCS, LCS, LCNS, and SCNS are NP-complete over a binary alphabet.

4 Consistent Sequences

In this section we show how the problem of finding sequences consistent with a pair of sets of strings can be related to the results of Section 2. Firstly, we make a simple observation considering the following sets of strings:

$$L' = \{0^n\} \qquad L'' = \{(1^m 0)^n 1^m\}$$

Clearly,

i) Every supersequence of L' contains at least n zeros.
ii) Every non-supersequence of L' contains at most $n-1$ zeros.
iii) Every subsequence of L'' contains at most n zeros. Every sequence with at most n zeros and at most m ones is a subsequence of L_2.
iv) Every non-subsequence of L'' containing at most m ones must contain at least $n+1$ zeros.

Relating this observation with the results of Section 2 we derive the following theorems.

Theorem 3 *Given a pair of sets of strings $\mathcal{L} = (L_1, L_2)$ over the alphabet $\{0,1\}$ with $|L_2| = 1$ the following problems are MAX SNP-hard:*

a) Find a shortest sequence of type $(Super, NSuper)$ or $(Super, Sub)$ consistent with $\mathcal{L}$.
b) Find a longest sequence of type $(Sub, NSub)$ or $(Sub, Super)$ consistent with $\mathcal{L}$.
c) Find a longest sequence of type $(NSuper, Super)$ or $(NSuper, NSub)$ consistent with $\mathcal{L}$.
d) Find a shortest sequence of type $(NSub, Sub)$ or $(NSub, NSuper)$ consistent with $\mathcal{L}$.

Proof. a) We make essentially the same reduction as in the proof of Theorem 2 a). To show the result for type $(Super, NSuper)$ we define $\mathcal{L} = (L_1, L_2)$ with $L_1 = L \cup \{0^{2n}\}$ and $L_2 = \{0^{2n+1}\}$ where L is the same as in the proof of Theorem 2 a). From the observation above we derive that every sequence of type $(Super, NSuper)$ consistent with $\mathcal{L}$ contains exactly $2n$ zeros. Now, the result follows from the proof of Theorem 2 a). Analogously, with the sets $L_1 = L \cup \{0^{2n}\}$ and $L_2 = \{(1^{4n+1}0)^{2n}1^{4n+1}\}$ we derive the result for type $(Super, Sub)$. Similarly, we derive b) to d) using the reductions of b) to d) in the proof of Theorem 2. □

Note, that the MAX SNP-hardness of finding a longest sequence of type $(NSuper, Super)$ consistent with a pair of sets of strings $\mathcal{L} = (L_1, L_2)$ over the alphabet $\{0,1\}$ with $|L_2| = 1$ has been shown previously by Zhang [10].

Theorem 4 *Given a pair of sets of strings $\mathcal{L} = (L_1, L_2)$ over the alphabet $\{0,1\}$ with $|L_2| = 2$ the following problems are NP-complete:*

a) Find a sequence of type $(Super, NSuper)$ or $(Super, Sub)$ consistent with $\mathcal{L}$.
b) Find a sequence of type $(Sub, NSub)$ or $(Sub, Super)$ consistent with $\mathcal{L}$.
c) Find a sequence of type $(NSuper, Super)$ or $(NSuper, NSub)$ consistent with $\mathcal{L}$.
d) Find a sequence of type $(NSub, Sub)$ or $(NSub, NSuper)$ consistent with $\mathcal{L}$.

Proof. We proceede similar to the proof of Theorem 3. □

Note, that the NP-completeness of finding a sequence of type $(Super, NSuper)$ consistent with a pair of sets of strings $\mathcal{L} = (L_1, L_2)$ over the alphabet $\{0, 1\}$ with $|L_2| = 2$ has been shown previously by Jiang and Li [2]. Note further, that the NP-completeness of finding a sequence of type $(Super, NSuper)$ consistent with $\mathcal{L}$ disproves a conjecture of Jiang and Li that was mentioned in the introduction (assuming P≠NP).

5 Conclusion

In this paper we studied the complexity of finding several kinds of common super- and subsequences for a given set of strings where the super- and subsequences are required to have a character composition that is (partially) fixed by the instance. We related these results with problems for finding sequences with respect to two given sets of strings. This approach allowed us to extend the results of several authors on such problems and characterize the complexity of these problems much clearer than has been done before. Moreover, we have shown that dual problems of SCS, LCS, LCNS are hard to approximize over a binary alphabet. Interesting problems remaining for future research are i) do SCS, LCS, LCNS, and SCNS have a polynomial time approximation scheme over a binary alphabet, ii) find an aproximation with a small factor for the MAX SNP-hard problems.

References

1. S. Arora, C. Lund, R. Motwani, R. Sundan, M. Szegedy, Proof verification and hardness of approximation problems, in: *Proc. 33rd IEEE Symp. on Foundations of Computer Science* (1992) 14-23.
2. T. Jiang, M. Li, On the complexity of learning strings and supersequences, *Theoret. Comput. Sci.* **119** (1993) 336-371.
3. D. Maier, The complexity of some problems on subsequences and supersequences, *J. ACM.* **25** (1978), 322-336.
4. M. Middendorf, The shortest common nonsubsequence Problem is NP-complete, *Theoret. Comput. Sci.* **108** (1993), 365-369.
5. C. Papadimitriou, M. Yannakakis, Optimization, Approximation and Complexity Classes, 20th ACM Symp. on Theory of Computing, (1988) 229-234.
6. L. Pitt, L.G. Valiant, Computational limitations on learning from examples. *J. ACM.* **35** (1988) 965-984.
7. K.-J. Räihä, E. Ukkonen, The shortest common supersequence problem over binary alphabet is NP-complete. *Theoret. Comput. Sci.* **16** (1981) 187-198.
8. V. G. Timkovsky, Complexity of common subsequence and supersequence problems and related problems, *Cybernetics* **25** (1990), 565-580.
9. L.G. Valiant, A theory of the learnable. *Comm. ACM.* **27** (1984) 1134-1142.
10. L. Zhang, The Approximation of Longest Common Non-Supersequences and Shortest Common Non-Subsequences is Hard, submitted for publication.