The Parameterized Complexity of Sequence Alignment and Consensus (Extended Abstract)*

Hans Bodlaender † Rodney G. Downey ‡ Michael R. Fellows §
H. Todd Wareham ¶

July 7, 2010

Abstract

The Longest common subsequence problem is examined from the point of view of parameterized computational complexity. There are several ways in which parameters enter the problem: the number of sequences to be analyzed, the length of the common subsequence, and the size of the alphabet. Lower bounds on the complexity of this basic problem imply lower bounds on more general sequence alignment and consensus problems. At issue in the theory of parameterized complexity is whether a problem can be solved in time $O(n^\alpha)$ for each fixed parameter value k, where α is a constant independent of k (termed *fixed-parameter tractability*). It can be argued that this is the appropriate asymptotic model of feasible computability for problems for which a small range of parameter values cover important applications — a situation which certainly holds for many problems in sequence analysis. Our main results show that: (1) The Longest Common Subsequence (LCS) parameterized by the number of sequences to be analyzed is hard for $W[t]$ for all t. (2) The LCS problem problem, parameterized by the length of the common subsequence, belongs to $W[P]$ and is hard for $W[2]$. (3) The LCS problem parameterized both by the number of sequences and the length of the common subsequence, is complete for $W[1]$. All of the above results

*to appear, Combinatorial Pattern Matching, Fifth Annual Conference, Asilomar, CA, June 1994.

†Computer Science Department, Utrecht University, P.O. Box 80.089, 3508 TB Utrecht, the Netherlands, hansb@cs.ruu.nl

‡Mathematics Department, Victoria University, P.O. Box 600, Wellington, New Zealand, downey@math.vuw.ac.nz

§Computer Science Department, University of Victoria, Victoria, British Columbia V8W 3P6, Canada, mfellows@csr.uvic.ca, *contact author*

¶Computer Science Department, Memorial University of Newfoundland, St. John's, Newfoundland A1C 5S7, Canada, harold@odie.cs.mun.ca

are for unrestricted alphabet sizes. For alphabets of a fixed size, problems (2) and (3) are fixed-parameter tractable. We conjecture that (1) remains hard.

1 Introduction

The computational problem of finding the longest common subsequence of a set of k strings (the LCS problem) has been studied extensively over the last twenty years (see [Hir83,IF92] and references). This problem has many applications. When $k = 2$, the longest common subsequence is a measure of the similarity of two strings and is thus useful in in molecular biology, pattern recognition, and text compression [San72,LF78,Mai78]. The version of LCS in which the the number of strings is unrestricted is also useful in text compression [Mai78], and is a special case of the multiple sequence alignment and consensus subsequence discovery problems in molecular biology [Pev92,DM93a,DM93b].

To date, most research has focused on deriving efficient algorithms for the LCS problem when $k = 2$ (see [Hir83,IF92] and references). Most of these algorithms are based on the dynamic programming approach [PM92], and require quadratic time. Though the k-unrestricted LCS problem is NP-complete [Mai78], certain of the algorithms for the $k = 2$ case have been extended to yield algorithms that require $O(n^{(k-1)})$ time and space, where n is the length of the longest of the k strings (see [IF92] and references; see also [Bae91]). Though such algorithms are polynomial-time for each fixed k, it would be interesting to know whether "truly" polynomial-time algorithms exist for each fixed k, i.e. does there an exist an algorithm for the k-fixed LCS problem with running time $O(f(k)n^c)$, where $f()$ is some function and c is a constant independent of k?

In this paper, we analyze the LONGEST COMMON SUBSEQUENCE problem from the point of view of parameterized complexity theory introduced in [DF92]. The parameterizations of the problem that we consider are defined as follows.

LONGEST COMMON SUBSEQUENCE (LCS-1, LCS-2 and LCS-3)
Instance: A set of k strings X_1, ..., X_k over an alphabet Σ, and a positive integer m.
Parameter 1: k (We refer to this problem as LCS-1.)
Parameter 2: m (We refer to this problem as LCS-2.)
Parameter 3: (k, m) (We refer to this problem as LCS-3.)
Question: Is there a string $X \in \Sigma^*$ of length at least m that is a subsequence of X_i for $i = 1, ..., k$?

In the §2 we give some background on parameterized complexity theory. In §3 we detail the proof that LCS-3 is complete for W[1]. This implies that LCS-1 and LCS-2 are W[1]-hard, results which can be improved by more elaborate arguments to show that LCS-1 is hard for $W[t]$ for all t, and that LCS-2 is hard for $W[2]$. Concretely, none of these three parameterized versions of LCS is thus fixed-parameter tractable unless the well-known (and apparently resistant) k-CLIQUE problem (and a host of others) are fixed-parameter tractable.

Problem	Fixed	Alphabet Size $\|\Sigma\|$	
		Unbounded	Fixed
LCS-1	k	W[t]-hard, $t \geq 1$	?
LCS-2	m	W[2]-hard	FPT
LCS-3	k, m	W[1]-complete	FPT

Table 1: The Parameterized Complexity of the LCS Problem.

Our results are summarized in the following table.

2 Parameterized Computational Complexity

The theory of parameterized compuational complexity is motivated by the observation that many NP-complete problems take as input two objects, for example, perhaps a graph G and and integer k. In some cases, e.g., k-Vertex cover and k-Min cut linear arrangement, the problem can be solved in linear time for every fixed k. For contrasting examples such as k-Clique, k-Dominating set and k-Bandwidth, the best known algorithms are based essentially on brute force, and require time $\Omega(n^k)$. If $P = NP$ then all of these problems are fixed-parameter tractable. The theory of parameterized computational complexity explores the apparent qualitative difference between these two classes of problems, and is particularly relevant to problems where a small range of parameter values covers important applications. This is certainly the case for many problems in computational biology. For these the theory offers a more sensitive view of tractability *vs.* (apparent) intractability than the theory of *NP*-completeness and may be a more appropriate complexity-analytic tool. The framework of the theory is sketched as follows.

Parameterized Problems, Fixed-Parameter Tractability and Reductions A *parameterized problem* is a set $L \subseteq \Sigma^* \times \Sigma^*$ where Σ is a fixed alphabet. For convenience, we consider that a parameterized problem L is a subset of $L \subseteq \Sigma^* \times N$. For a parameterized problem L and $k \in N$ we write L_k to denote the associated fixed-parameter problem $L_k = \{x|(x,k) \in L\}$. We say that a parameterized problem L is (uniformly) *fixed-parameter tractable* if there is a constant α and an algorithm Φ such that Φ decides if $(x,k) \in L$ in time $f(k)|x|^\alpha$ where $f : N \to N$ is an arbitrary function. Where A and B are parameterized problems, we say that A is (uniformly many:1) *reducible* to B if there is an algorithm Φ which transforms (x,k) into $(x', g(k))$ in time $f(k)|x|^\alpha$, where $f, g : N \to N$ are arbitrary functions and α is a constant independent of k, so that $(x,k) \in A$ if and only if $(x', g(k)) \in B$.

Complexity Classes The classes of the W hierarchy are based intuitively on the complexity of the circuits required to check solutions. A Boolean circuit defined to be of *mixed type* if it consists of circuits having gates of the following kinds: (1) *Small gates*: *not* gates, *and* gates and *or* gates with bounded fan-in. (2) *Large gates*: *and* gates and *or* gates with

unrestricted fan-in. The *depth* of a circuit C is defined to be the maximum number of gates (small or large) on an input-output path in C. The *weft* of a circuit C is the maximum number of large gates on an input-output path in C. We say that a family of decision circuits F has *bounded depth* if there is a constant h such that every circuit in the family F has depth at most h. We say that F has *bounded weft* if there is constant t such that every circuit in the family F has weft at most t. The *weight* of a boolean vector x is the number of 1's in the vector.

Definition. Let F be a family of decision circuits. We allow that F mayPha2 QQJvpn—Pdifferent circuits with a given number of inputs. To F we associate the parameterized circuit problem $L_F = \{(C, k) : C \text{ accepts an input vector of weight } k\}$. A parameterized problem L belongs to $W[t]$ if L reduces to the parameterized circuit problem $L_{F(t,h)}$ for the family $F(t, h)$ of mixed type decision circuits of weft at most t, and depth at most h, for some constant h. A parameterized problem L belongs to $W[P]$ if L reduces to the circuit problem L_F, where F is the set of all circuits (no restrictions). We designate the class of fixed-parameter tractable problems FPT.

$$FPT \subseteq W[1] \subseteq W[2] \subseteq \cdots \subseteq W[P]$$

All of the following problems are now known to be complete for $W[1]$: Square tiling, Independent set, Clique, and Bounded post correspondence problem, k-Step derivation for context-sensitive grammars, Vapnik-Chervonenkis dimension, k-Step halting problem for nondeterministic Turing machines [CCDF93, DEF93, DFKHW93]. Thus, any one of these problems is fixed-parameter tractable if and only if all of the others are.

3 The Reductions

In general, issues in parameterized complexity tend to be more difficult to resolve than corresponding issues in traditional (e.g. *NP*-completeness) complexity analysis. The reductions by which our main theorems are established are quite complicated and can only be sketched in this abstract.

Theorem 1. LCS-3 is complete for $W[1]$.

Proof Sketch. Membership in $W[1]$ can be seen by a reduction to Weighted cnf satisfiability for expressions having bounded clause size. The idea is to use a truth assignment of weight k^2 to indicate the k positions in each of the k strings of an instance of LCS-3 that yield a common subsequence of length k. Details are omitted for this abstract.

To show $W[1]$-hardness we reduce from k-Clique. Let $G = (V, E)$ be a graph for which we wish to determine whether G has a k-clique. We show how to construct a family $\mathcal{F}_G$ of $k' = f(k)$ sequences over an alphabet Σ that have a common subsequence of length $k'' = g(k)$ if and only G contains a k-clique. Assume for convenience that the vertex set of

G is $V = \{1, \ldots, n\}$.

The Alphabet We first describe the alphabet $\Sigma = \Sigma_1 \cup \Sigma_2 \cup \Sigma_3 \cup \Sigma_4$. We refer to these as *vertex symbols* (Σ_1), *edge symbols* (Σ_2), *vertex position symbols* (Σ_3), and *edge position symbols* (Σ_4).

$$\Sigma_1 = \{\alpha[p,q,r] : \ 1 \le p \le k, \ 0 \le q \le 1, \ 1 \le r \le n\}$$

$$\Sigma_2 = \{\beta[i,j,q,u,v] : \ 1 \le i < j \le k, \ 0 \le q \le 1, \ 1 \le u < v \le n, \ uv \in E\}$$

$$\Sigma_3 = \{\gamma[p,q,b] : \ 1 \le p \le k, \ 0 \le q \le 1, 0 \le b \le 1\}$$

$$\Sigma_4 = \{\delta[i,j,q,b] : \ 1 \le i < j \le k, \ 0 \le q \le 1, \ 0 \le b \le 1\}$$

We will use the following shorthand notation to refer to various subsets of Σ. The notation indicates which indices are held fixed to some value, with the symbol * indicating that the index should vary over its range of definition in building the set. For example, $\Sigma_1[p,*,r] = \{\alpha[p,q,r] : \ 0 \le q \le 1\}$ is the set of two elements with the first and third indices fixed at p and r, respectively.

The Target Parameters There are $f_1(k) = 2k + k(k-1) = k^2 + k$ position symbols (in Σ_3 and Σ_4). We take $w = f_1(k)^2 + 1$, $k' = f_1(k) + 2$, and $k'' = (w+1)f_1(k)$.

Symbol Subsets and Operations It is convenient to introduce a linear ordering on Σ that corresponds to the "natural" order in which the various symbols occur, as illustrated by the example above. We can achieve this by defining a "weight" on the symbols of Σ and then ordering the symbols by weight.

Let $N = 2kn$ (a value conveniently larger than k and n). Define the *weight* $||a||$ of a symbol $a \in \Sigma$ by

$$||a|| = \begin{cases} pN^6 + qN^5 + r & \text{if } \ a = \alpha[p,q,r] \in \Sigma_1 \\ q'iN^6 + qjN^6 + q'N^4 + q'jN^3 + qiN^3 + uN + v & \text{if } \ a = \beta[i,j,q,u,v] \in \Sigma_2 \\ pN^6 + qN^5 + bN^2 & \text{if } \ a = \gamma[p,q,b] \in \Sigma_3 \\ q'iN^6 + qjN^6 + q'N^4 + q'jN^3 + qiN^3 + bN^2 & \text{if } \ a = \delta[i,j,q,b] \in \Sigma_4 \end{cases}$$

where $q' = (q-1)^2$.

Define a linear order on Σ by $a < b$ if and only if $||a|| < ||b||$. The reader can verify that, assuming $a < b < c$, the symbols of the example sequence $\sigma(a,b,c)$ described above occur in ascending order.

For $a, b \in \Sigma$, $a < b$, we define the *segment* $\Sigma(a,b)$ to be $\Sigma(a,b) = \{e \in \Sigma : a \le e \le b\}$, and we define similarly the segments $\Sigma_i(a,b)$.

If Γ is a finite set of symbols, then it is easy to see that there is a "universal" string $(m\Gamma) \in \Gamma^*$ of length $m|\Gamma|$ that contains as a subsequence every string of length at most m ove9 *.avma*, for example, by running through the symbols in Γ m times. We will use the notation $(m\Gamma)$ to refer to any choice of such a string. Where m is unimportant except that

it be "large enough" (with the understanding that this means also "not too large") we may write $(*\Gamma)$ for convenience.

If $\Gamma \subseteq \Sigma$, let $(\uparrow \Gamma)$ be the string of length $|\Gamma|$ which consists of one occurence of each symbol in Γ in ascending order, and let $(\downarrow \Gamma)$ be the string of length $|\Gamma|$ which consists of one occurence of each symbol in Γ in descending order.

String Gadgets We next describe some "high level" component subsequences for the construction. In the following let $\updownarrow$ denote either $\uparrow$ or $\downarrow$. Product notation is interpreted as refering to concatenation.

Vertex and Edge Selection Gadgets

$$\langle \updownarrow \text{ vertex } p\rangle = \gamma[p,0,0]^w(\updownarrow \Sigma_1[p,0,*])\gamma[p,0,1]^w$$

$$\langle \updownarrow \text{ vertex } p \text{ echo}\rangle = \gamma[p,1,0]^w(\updownarrow \Sigma_1[p,1,*])\gamma[p,1,1]^w$$

$$\langle \updownarrow \text{ edge } (i,j)\rangle = \delta[i,j,0,0]^w(\updownarrow \Sigma_2[i,j,0,*,*])\delta[i,j,0,1]^w$$

$$\langle \updownarrow \text{ edge } (i,j) \text{ echo}\rangle = \delta[i,j,1,0]^w(\updownarrow \Sigma_2[i,j,1,*,*])\delta[i,j,1,1]^w$$

$$\langle \updownarrow \text{ edge } (i,j) \text{ from } u\rangle = \delta[i,j,0,0]^w(\updownarrow \Sigma_2[i,j,0,u,*])\delta[i,j,0,1]^w$$

$$\langle \updownarrow \text{ edge } (i,j) \text{ to } v\rangle = \delta[i,j,1,0]^w(\updownarrow \Sigma_2[i,j,1,*,v])\delta[i,j,1,1]^w$$

Control and Selection Assemblies

$$\begin{aligned}\langle \updownarrow \text{ control } p\rangle &= \langle \updownarrow \text{ vertex } p\rangle \left(\prod_{s=1}^{p-1}\langle \updownarrow \text{ edge } (s,p) \text{ echo}\rangle\right) \\ &\quad \cdot \left(\prod_{s=p+1}^{k}\langle \updownarrow \text{ edge } (p,s)\rangle\right)\langle \updownarrow \text{ vertex } p \text{ echo}\rangle\end{aligned}$$

$$\begin{aligned}\langle \uparrow \text{ choice } p\rangle &= \prod_{x=1}^{n}\Bigg(\gamma[p,0,0]^w\alpha[p,0,x]\gamma[p,0,1]^w\prod_{t=1}^{p-1}\langle \uparrow \text{ edge } (t,p) \text{ to } x\rangle \\ &\quad \cdot \prod_{t=p+1}^{k}\langle \uparrow \text{ edge } (p,t) \text{ from } x\rangle\gamma[p,1,0]^w\alpha[p,1,x]\gamma[p,1,1]^w\Bigg)\end{aligned}$$

$$\begin{aligned}\langle \downarrow \text{ choice } p\rangle &= \prod_{x=n}^{\text{down to } 1}\Bigg(\gamma[p,0,0]^w\alpha[p,0,x]\gamma[p,0,1]^w\prod_{t=1}^{p-1}\langle \downarrow \text{ edge } (t,p) \text{ to } x\rangle \\ &\quad \cdot \prod_{t=p+1}^{k}\langle \downarrow \text{ edge } (p,t) \text{ from } x\rangle\gamma[p,1,0]^w\alpha[p,1,x]\gamma[p,1,1]^w\Bigg)\end{aligned}$$

Edge Symbol Pairing Gadget

$$\langle \text{edge } (i,j) \text{ from } u \text{ to } v\rangle = \beta[i,j,0,u,v]\,(*\Sigma(\delta[i,j,0,1],\delta[i,j,1,0]))\,\beta[i,j,1,u,v]$$

The Reduction We may now describe the reduction. The instance of LCS-3 consists of strings which we may consider as belonging to three subsets: *Control*, *Selection* and *Check*. The two strings in the *Control* set are

$$X_1 = \prod_{t=1}^{k} \langle \uparrow \text{control } t\rangle$$

$$X_2 = \prod_{t=1}^{k} \langle \downarrow \text{control } t\rangle$$

The $2k$ strings in the *Selection* set are, for $p = 1, ..., k$

$$Y_p = \left(\prod_{t=1}^{p-1} \langle \uparrow \text{control } t\rangle\right) \langle \uparrow \text{choice } p\rangle \left(\prod_{t=p+1}^{k} \langle \uparrow \text{control } t\rangle\right)$$

$$Y'_p = \left(\prod_{t=1}^{p-1} \langle \downarrow \text{control } t\rangle\right) \langle \downarrow \text{choice } p\rangle \left(\prod_{t=p+1}^{k} \langle \uparrow \text{control } t\rangle\right)$$

The $2\binom{k}{2} = k(k-1)$ strings in the *Check* set are, for $1 \le i < j \le k$

$$\begin{aligned} Z_{i,j} \;=\; & \left(\prod_{t=1}^{i-1} \langle \uparrow \text{control } t\rangle\right) \langle \uparrow \text{vertex } i\rangle \left(\prod_{s=1}^{i-1} \langle \uparrow \text{edge } (s,i) \text{ echo}\rangle\right) \left(\prod_{s=i+1}^{j-1} \langle \uparrow \text{edge } (i,s)\rangle\right) \\ & \cdot\; \delta[i,j,0,0]^w \prod_{\substack{1 \le u < v \le n \\ uv \in E}}^{\text{lex}\uparrow} \langle \text{edge } (i,j) \text{ from } u \text{ to } v\rangle \\ & \cdot\; \delta[i,j,1,1]^w \left(\prod_{s=i+1}^{j-1} \langle \uparrow \text{edge } (s,j) \text{ echo}\rangle\right) \left(\prod_{s=j+1}^{k} \langle \uparrow \text{edge } (j,s)\rangle\right) \\ & \cdot\; \langle \uparrow \text{vertex } j \text{ echo}\rangle \langle \prod_{t=j+1}^{k} \langle \uparrow \text{control } t\rangle \end{aligned}$$

$$\begin{aligned} Z'_{i,j} \;=\; & \left(\prod_{t=1}^{i-1} \langle \downarrow \text{control } t\rangle\right) \langle \downarrow \text{vertex } i\rangle \left(\prod_{s=1}^{i-1} \langle \downarrow \text{edge } (s,i) \text{ echo}\rangle\right) \left(\prod_{s=i+1}^{j-1} \langle \downarrow \text{edge } (i,s)\rangle\right) \\ & \cdot\; \delta[i,j,0,0]^w \prod_{\substack{1 \le u < v \le n \\ uv \in E}}^{\text{lex}\downarrow} \langle \text{edge } (i,j) \text{ from } u \text{ to } v\rangle \end{aligned}$$

$$\cdot \quad \delta[i,j,1,1]^w \left(\prod_{s=i+1}^{j-1} \langle \downarrow \text{ edge } (s,j) \text{ echo} \rangle \right) \left(\prod_{s=j+1}^{k} \langle \downarrow \text{ edge } (j,s) \rangle \right)$$

$$\cdot \quad \langle \downarrow \text{ vertex } j \text{ echo} \rangle \langle \prod_{t=j+1}^{k} \langle \downarrow \text{ control } t \rangle$$

We comment that the key difference between $Z_{i,j}$ and $Z'_{i,j}$ is that in $Z_{i,j}$ the edge symbol pairing gadgets occur in increasing lexicographic order, and in $Z'_{i,j}$ the gadgets are in decreasing lexicographic order.

Proof of Correctness Where S_1 and S_2 are strings of symbols, let $l(S_1, S_2)$ denote the maximum length of a common subsequence of S_1 and S_2. In the Control Strings X_1 and X_2 we distinguish certain substrings that we term *positions*. Note that both of these strings are formed as the concatenation of four different kinds of substrings: $\langle$vertex$\rangle$, $\langle$vertex echo$\rangle$, $\langle$edge$\rangle$ and $\langle$edge echo$\rangle$, and that each of these "vertex and edge selection" substrings begins and ends with a matched pair of substrings of repeated symbols from Σ_3 (in the case of vertex selection), or from Σ_4 (in the case of edge selection). These matched pairs of position symbol substrings determine a *position* — note that these position symbol substrings (and therefore the positions defined) occur in the same order in X_1 and X_2. Thus there are $k(2+k-1) = k^2 + k$ positions.

Between a matched pair of position symbol substrings in X_1 there is a set of symbols in increasing order that we will term a *set of (vertex or edge) stairs*, and in X_2 in the corresponding position there occurs the same set of symbols in decreasing order. The proof of the following Claim 1 is trivial, and the proof of Claim 2 is omitted for reasons of space.

Claim 1. Suppose Σ is a linearly ordered finite alphabet, and that $S\uparrow$ is the string consisting of the symbols of Σ in increasing order, and that $S\downarrow$ is the symbols of Σ in decreasing order. Then $l(S\uparrow, S\downarrow) = 1$. □

Claim 2. A common subsequence C of the control sequences X_1 and X_2 of maximum length l satisfies the conditions: (1) $l = k''$, and (2) C consists of the position symbol substrings (common to X_1 and X_2) together with one symbol in each position defined by these substrings.

By Claim 2, if C is a common vertex of X_1 and X_2 of length k'', we may refer unambiguously to the vertices and edges represented in the various positions of C. In particular, note that these positions occur in k *vertex units*, each of which consists of an *initial vertex position*, followed by $k-1$ *edge* and *edge echo positions* and concluding with a *terminal vertex echo position*. If uv is an edge of the graph with $u < v$, then we refer to u as the *initial vertex* and to v as the *terminal vertex* of the edge. The following can be proved.

Claim 3. If C is a subsequence of length k'' common to the Control and Selection sets, then in each vertex unit: (1) the vertex u represented in the initial vertex position is also represented in the terminal vertex echo position, (2) each edge represented in an edge echo

position has terminal vertex u, and (3) each edge represented in an edge position has initial vertex u.

The length w substrings of the position symbols $\delta[i,j,0,0]$ and $\delta[i,j,0,1]$ in C define the $(i,j)^{th}$ edge position in the i^{th} vertex unit and the length w substrings of the position symbols $\delta[i,j,1,0]$ and $\delta[i,j,1,1]$ in C define the $(i,j)^{th}$ edge echo position in the j^{th} vertex unit. We term these a *corresponding pair* of edge and edge echo positions.

Claim 4. If C is a subsequence of length k'' common to the Control, Selection and Check sets, then for each corresponding pair of an edge position and an edge echo position, the same edge must be represented in the two positions.

Proof. Suppose C is a subsequence of length k'' common to the Control and Selection sets. We argue that if C is also common to $Z_{i,j}$ and $Z'_{i,j}$ then Lemma holds for the $(i,j)^{th}$ corresponding pair of positions. Let $C_{i,j}$ and $C'_{i,j}$ denote specific subsequences of $Z_{i,j}$ and $Z'_{i,j}$ isomorphic to C.

It is convenient to consider $Z_{i,j}$ (and similarly $Z'_{i,j}$) under the factorization $Z_{i,j} = Z_{i,j}(1)Z_{i,j}(2)Z_{i,j}(3)$ where

$$Z_{i,j}(2) = \prod_{\substack{1 \le u < v \le n \\ uv \in E}}^{\text{lex}\uparrow} \langle \text{edge } (i,j) \text{ from } u \text{ to } v \rangle$$

and where $Z_{i,j}(1)$ and $Z_{i,j}(3)$ are the appropriately defined prefix and suffix (respectively) of $Z_{i,j}$.

Since none of the position symbols in $Z_{i,j}(1)$ or $Z_{i,j}(3)$ occur in $Z_{i,j}(2)$, all of the position symbols in $Z_{i,j}(1)$ and $Z_{i,j}(3)$ must belong to $C_{i,j}$. Similarly, all of the position symbols in $Z'_{i,j}(1)$ and $Z'_{i,j}(3)$ must belong to $C'_{i,j}$. This implies, by Lemma 2, that $C_{i,j} \cup Z_{i,j}(2) = C'_{i,j} \cup Z'_{i,j}(2)$ begins with a symbol $\beta[i,j,0,u,v]$ and ends with a symbol $\beta[i,j,0,x,y]$. We argue that necessarily $u = x$ and $v = y$.

¿From the fact that $\beta[i,j,1,x,y]$ follows $\beta[i,j,0,u,v]$ in $Z_{i,j}(2)$, and from the construction of the latter in increasing lexicographic order, we may deduce that (u,v) precedes (x,y) lexicographically. Similarly, since $Z'_{i,j}(2)$ is constructed in decreasing lexicographic order, we obtain that (x,y) precedes (u,v), and therefore $(x,y) = (u,v)$. □

We now argue the correctness of the reduction as follows. If G has a k-clique, then it is easily seen that there is a common subsequence of length k'' in which the k vertex units represent the vertices of the clique, and the edge and edge echo positions within each vertex unit represent the edges incident on the represented vertex of the unit in increasing lexicographic order. (Each edge is thus represented twice, in the vertex units corresponding to its endpoints, first in an edge position in the initial vertex unit, and second in an edge echo position in the terminal vertex unit.)

Conversely, suppose there is a common subsequence C of length k''. By Claims 2 and

3, C represents a sequence of k vertices of G. That these must be a clique in G follows from Claim 4 and the definition of the "edge from" and "edge to" gadgets, which restrict the edges represented in a vertex unit to those present in the graph and for which the vertex is, respectively, initial or terminal. That completes the proof. □

Theorem 1 implies immediately that LCS-1 and LCS-2 are hard for $W[1]$. We can strengthen this result by more complicated reductions that space limitations preclude describing in this abstract.

Theorem 2. LCS-1 is hard for $W[t]$ for every positive integer t.

Theorem 3. LCS-2 is hard for $W[2]$, and belongs to $W[P]$.

4 Conclusions

Our results have implications for the fixed-parameter tractability of the multiple sequence alignment and consensus subsequence discovery problems in molecular biology because the LCS problem is a special case of each of these problems.

One weakness of our results is that we consider above only the case of unbounded alphabet sizes. When the size of the alphabet is a fixed constant, it is easy to observe that LCS-2 (and thus also LCS-3) are fixed-parameter tractable. We conjecture, but do not yet have a proof, that LCS-1 remains hard for $W[t]$ for all t, for alphabets of size at least 2.

References

[Bae91] R. A. Baeza-Yates, "Searching Subsequences", *Theoretical Computer Science*, 78, 363–376, 1991.

[CCDF93] L. Cai, J. Chen, R. Downey and M. Fellows, "The Parameterized Complexity of Short Computations and Factorizations," University of Victoria, Technical Report, Department of Computer Science, July, 1993.

[DEF93] R. Downey, P. Evans and M. Fellows, "Parameterized Learning Complexity," *Proc. Sixth ACM Workshop on Computational Learning Theory (COLT)*, pp. 51–57, ACM Press, 1993.

[DF92] R. Downey and M. Fellows, "Fixed-Parameter Intractability (Extended Abstract)". In *Proceedings of the Seventh Annual Conference on Structure in Complexity Theory*, pp. 36–49, IEEE Computer Society Press, Los Alamitos, CA, 1992.

[DFKHW93] R. Downey, M. Fellows, B. Kapron, M. Hallett and T. Wareham, "The Parameterized Complexity of Some Problems in Logic and Linguistics," Workshop on Recursion Theory and Complexity in Logic, Vancouver, B.C., Canada, October, 1993, and University

of Victoria, Technical Report, Department of Computer Science, July, 1993.

[DM93a] W. H. E. Day and F. R. McMorris, "Discovering Consensus Molecular Sequences". In O. Opitz, B. Lausen, and R. Klar (eds.) *Information and Classification – Concepts, Methods, and Applications*, pp. 393–402, Springer-Verlag, Berlin, 1993.

[DM93b] W. H. E. Day and F. R. McMorris, "The Computation of Consensus Patterns in DNA Sequences", *Mathematical and Computer Modelling*, 17(10), 49–52, 1993.

[Hir83] D. S. Hirschberg, "Recent Results on the Complexity of Common Subsequence Problems". In D. Sankoff and J. B. Kruskal (eds.) *Time Warps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison*, pp. 325–330, Addison-Wesley, Reading, MA, 1983.

[IF92] R. W. Irving and C. B. Fraser, "Two Algorithms for the Longest Common Subsequence of Three (or More) Strings". In A. Apostolico, M. Crochemore, Z. Galil, and U. Manber (eds.) *Proceedings of the Third Annual Symposium on Combinatorial Pattern Matching*, pp. 214–229, Lecture Notes in Computer Science no. 644, Springer-Verlag, Berlin, 1992.

[LF78] S. Y. Lu and K. S. Fu, "A Sentence-to-Sentence o Clustering Procedure for Pattern Analysis", *IEEE Transactions on Systems, Man, and Cybernetics*, 8, 381–389, 1978.

[Mai78] D. Maier, "The Complexity of Some Problems on Subsequences and Supersequences", *Journal of the ACM*, 25(2), 322–336, 1978.

[PM92] W. R. Pearson and W. Miller, "Dynamic Programming Algorithms for Biological Sequence Comparison", *Methods in Enzymology*, 183, 575–601, 1992.

[Pev92] P. A. Pevzner, "Multiple Alignment, Communication Cost, and Graph Matching", *SIAM Journal on Applied Mathematics*, 52(6), 1763-1779, 1992.

[San72] D. Sankoff, "Matching Comparisons under Deletion/Insertion Constraints", *PNAS*, 69(1), 4–6, 1972.