Parameterized Complexity Analysis in Computational Biology

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

April 20, 1995

Abstract

Many computational problems in biology involve parameters for which a small range of values cover important applications. We argue that for many problems in this setting, parameterized computational complexity rather than NP-completeness, is the appropriate tool for studying apparent intractability. 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, where α is a constant independent of the parameter. In addition to surveying this complexity framework, we describe some new results on the LONGEST COMMON SUBSEQUENCE problem. In particular, we show that the problem is hard for $W[t]$ for all t when parameterized by the number of strings and the size of the alphabet. Lower bounds on the complexity of this basic combinatorial problem imply lower bounds on more general sequence alignment and consensus discovery problems. We also describe a number of open problems pertaining to the parameterized complexity of problems in computational biology where small parameter values are important.

Keywords: Multiple Alignment of Genetic Sequences; Consensus Sequences; Longest Common Subsequence; Parameterized Complexity.

*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, University of Victoria, Victoria, British Columbia V8W 3P6, Canada, mhallett@csr.uvic.ca

¶Computer Science Department, University of Victoria, Victoria, British Columbia V8W 3P6, Canada, harold@csr.uvic.ca

Parameterized Complexity Analysis in Computational Biology

Abstract

Many computational problems in biology involve parameters for which a small range of values cover important applications. We argue that for many problems in this setting, parameterized computational complexity rather than NP-completeness, is the appropriate tool for studying apparent intractability. 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, where α is a constant independent of the parameter. In addition to surveying this complexity framework, we describe some new results on the LONGEST COMMON SUBSEQUENCE problem. In particular, we show that the problem is hard for $W[t]$ for all t when parameterized by the number of strings and the size of the alphabet. Lower bounds on the complexity of this basic combinatorial problem imply lower bounds on more general sequence alignment and consensus discovery problems. We also describe a number of open problems pertaining to the parameterized complexity of problems in computational biology where small parameter values are important.

Keywords: Multiple Alignment of Genetic Sequences; Consensus Sequences; Longest Common Subsequence; Parameterized Complexity.

1 Introduction

The theory of parameterized computational complexity introduced in [DF92] is designed to address a natural and important qualitative complexity distinction which lies "beyond NP-completeness." In order to make the difference as clear as possible, the reader may assume that every problem discussed in this paper is NP-complete, and thus NP-completeness theory has said all that it *can* say about any of them.

We are furthermore interested throughout this paper in computational problems for which the input consists of two parts, one of which is termed the *parameter* for the problem. For illustration, we might consider the following two well-known computational problems concerning graphs. Each of them takes as input a graph $G = (V, E)$ and a positive integer k, and in each case we consider the parameter to be k.

VERTEX COVER: Is there a set $V' \subseteq V$ of k vertices such that for every edge $uv \in E$, either $u \in V'$ or $v \in V'$?

DOMINATING SET: Is there a set $V' \subseteq V$ of k vertices such that for every vertex $u \in V$ there is an edge $uv \in E$ with $v \in V'$?

Despite their superficial similarity, the first is solvable in linear time for each fixed k, and the algorithm is practical for $k \leq 20$. The best known algorithm for the second problem requires times $O(n^{k+1})$ and is no more clever than simply trying all possible k-element sets of vertices. We now have some evidence that DOMINATING SET is qualitatively more difficult. In [DF92] it is shown that this (parameterized) problem is complete for the parameterized complexity class W[2] (see §2), and thus is unlikely to be solvable in time $O(n^{\alpha})$ for each fixed parameter value, and for some constant α independent of k. (For VERTEX COVER we have $\alpha = 1$.)

In a wide variety of settings, computational problems arise for which a small range of parameter values cover many important applications; one purpose of this paper is to point out a number of these in computational biology (see §4). In these situations, NP-completeness can be far too

pessimistic; the tool of choice for elucidating inherent problem difficulties is parameterized complexity analysis. Several recent papers have applied this theory to problems in biological computing [BFW92, BFH93, FHW93]. We wish to make the point that the theory is potentially of very wide applicability in computational biology.

In §2 the basics of parameterized complexity theory are briefly reviewed. In §3 we describe some new results on the LONGEST COMMON SUBSEQUENCE (LCS) problem, where the size of the alphabet is taken into account. In §4 we describe a number of parameterized problems in biological computing where this sort of complexity analysis would seem to be appropriate.

2 Parameterized Computational Complexity

The framework of the theory is sketched as follows (for greater detail see [DF92]).

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 may have many different 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, Bounded post correspondence problem, k-Step derivation for context-sensitive grammars, Vapnik-Chervonenkis dimension, and the 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 Multiple Sequence Alignment and Consensus

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 [IF92] and references). The k-unrestricted LCS problem is NP-complete even if the alphabet is of size two or greater [Mai78], and the best known algorithms require $O(n^{(k-1)})$ time and space (see [IF92] and references). Our interest in this problem comes from the fact that it is a special case of some formulations of

multiple sequence alignment and consensus subsequence discovery problems [Pev92, DM93]. Thus complexity lower bounds on the LCS problem imply complexity lower bounds on several more complex and realistically formulated problems in biological computing.

Our main concern in this section is with the complexity of the following three parameterized versions of LCS.

LONGEST COMMON SUBSEQUENCE (LCS-A, LCS-B)

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-A.)

Parameter 2: $k, |\Sigma|$ (We refer to this problem as LCS-B.)

Question: Is there a string $X \in \Sigma^*$ of length at least m that is a subsequence of X_i for $i = 1, ..., k$?

By LCS-C we refer to LCS-A where the size of the alphabet Σ is a fixed constant.

All of these parameterized problems are relevant because the number of input strings to the associated biological problems will usually be between 3 and 12. The most compelling of these problems is LCS-C, since the alphabet for biological sequences is often of fixed constant size, e.g. DNA and protein sequences have alphabets of size 4 and 20, respectively. Problem LCS-B can be viewed as a kind of approximation to LCS-C. Our failure to find a hardness result for LCS-C invites hope that it could be fixed-parameter tractable.

Theorem. LCS-A and LCS-B are hard for $W[t]$ for all t.

Proof. The proof consists of the composition of two reductions, the first to LCS-A, and the second from LCS-A to LCS-B.

The First Reduction

By the results of [DF92] we may take the source instance of the reduction to be a t-normalized expression E and a positive integer k, where t is even and E is monotone. Let n denote the number

of variables of E. By simple padding we may assume that E has the form:

$$E = \wedge_{i_1=1}^{n} \vee_{i_2=1}^{n} \cdots \wedge_{i_{t-1}=1}^{n} \vee_{i_t=1}^{n} l[i_1, \ldots, i_t]$$

Let $V = \{u_1, ..., u_n\}$ denote the set of variables of E. Thus in the above expression for E, $l[i_1, \ldots, i_t]$ is always a positive literal, that is, an element of V. We show how to produce an instance of LCS-A consisting of $\binom{k}{2} + 2k + 2$ strings that have a common subsequence of length m if and only if E has a weight k truth assignment, with m described:

$$m = 3k + 3n^{t/2} + 2\sum_{j=0}^{t} c(j)w(j,t)$$

where

$$c(j) = n^{\lceil j/2 \rceil}$$

and

$$w(j,t) = n^{2t(t-j)}$$

We will use the following notation for indexing. The set $\{1, ..., n\}$ is denoted as $[n]$. By $[n]^r$ we mean the set $\{\alpha = (a_1, ..., a_r) : \ 1 \leq a_i \leq n \text{ for } 1 \leq i \leq r\}$. By $[n]^0$ we denote the singleton set $\{\epsilon\}$ where ϵ denotes the unique vector of length 0 over $[n]$. If $\alpha \in [n]^s$ and $b \in [n]$ with $\alpha = (a_1, ..., a_s)$, then we write $\alpha.b$ to denote $(a_1, ..., a_s, b) \in [n]^{s+1}$. We consider that $[n]^r$ is ordered lexicographically, and will use $\uparrow$ lex to denote increasing lexicographic order and $\downarrow$ lex to denote decreasing lexicographic order. We make use of the index set I defined

$$I = \bigcup_{r=1}^{t} [n]^r$$

We say that $\alpha \in I$ is *even* if $\alpha \in [n]^r$ for r even, otherwise α is termed *odd*. If $\alpha \in [n]^r$ then we write $|\alpha| = r$ and term this the *rank* of α. If $\alpha, \beta \in I$ and α is a proper prefix of β then we write $\alpha \prec \beta$.

The Alphabet

The alphabet Σ for target instance of LCS-A can be expressed as the union

$$\Sigma = \Sigma_1 \cup \Sigma_2 \cup \Sigma_3 \cup \Sigma_4 \cup \Sigma_5$$

where

$$\begin{aligned}
\Sigma_1 &= \{c[j], c'[j] : \ 1 \leq j \leq k\} \\
\Sigma_2 &= \{v[i,j] : \ 1 \leq i \leq n, \ 1 \leq j \leq k\} \\
\Sigma_3 &= \{p[\alpha], p'[\alpha] : \ \alpha \in I\} \\
\Sigma_4 &= \{q[\alpha, j], q'[\alpha, j] : \ \alpha \in [n]^t, \ 1 \leq j \leq k\} \\
\Sigma_5 &= \{u[\alpha, i, j] : \ \alpha \in [n]^t, \ 1 \leq i \leq n, \ 1 \leq j \leq k\}
\end{aligned}$$

Symbol Subsets, Order and Rank Let Σ' denote $\Sigma_1 \cup \Sigma_2 \cup \Sigma_4 \cup \Sigma_5$. If S_1 and S_2 are sets of symbols of an ordered alphabet, then $S_1 < S_2$ denotes that for all $a \in S_1$ and $b \in S_2$, $a < b$. We consider that Σ' is linearly ordered in the unique way consistent with the following requirements:

- $(\Sigma_1 \cup \Sigma_2) < (\Sigma_4 \cup \Sigma_5)$
- Σ_2 is ordered lexicographically by symbol index.
- For all $i \in [n]$ and $j \in [k]$, $c[j] < v[i,j] < c'[j]$.
- For $1 \leq i < j \leq k$, $\{c[i], c'[i]\} < \{c[j], c'[j]\}$.
- If (α, j) precedes (β, h) lexicographically, then $\{q[\alpha, j], q'[\alpha, j]\} < \{q[\beta, h], q'[\beta, h]\}$.
- Σ_5 is ordered lexicograpically by symbol index.
- $q[\alpha, j] < u[\alpha, i, j] < q'[\alpha, j]$ for all $\alpha \in [n]^t$, $i \in [n]$ and $j \in [k]$.

By $\Sigma'[a, b]$ we denote the set of symbols $\{s \in \Sigma' : a \leq s \leq b\}$ in the above linear ordering.

Each of the symbols in $\Sigma'' = \Sigma_3 \cup \Sigma_4 \cup \Sigma_5$ is (partially) indexed by some $\alpha \in I$. We term the *rank* of a symbol s in this set, denoted $|s|$, to be the rank $|\alpha|$ of the index α. If $\Gamma \subseteq \Sigma$ is a set of symbols, then $\Sigma''[r]$ denotes the set of symbols in Γ of rank r, $0 \leq r \leq t$.

In discussing strings over the alphabet Σ, if $\Gamma \subseteq \Sigma$ is a symbol subset and $S \in \Sigma^*$, then by $S \cap \Gamma$ we denote the subsequence of S consisting of all symbols in Γ. We write $|S|$ to denote the length of a string S.

Substring Gadgets Product notation in the description of these components refers to string concatenation. Where s is a symbol, the notation s^w denotes the symbol s repeated w times. Note that in some cases products are formed in decreasing order according to some index, which is indicated by notation such as

$$\prod_{i=n}^{1} \cdots$$

The following strings provide gadgets for our reduction.

$$\langle\uparrow \text{ selection } j\rangle = c[j]\left(\prod_{i=1}^{n} v[i,j]\right)c'[j]$$

$$\langle\downarrow \text{ selection } j\rangle = c[j]\left(\prod_{i=n}^{1} v[i,j]\right)c'[j]$$

$$\langle\uparrow \text{ select } \rangle = \prod_{j=1}^{k}\langle\uparrow \text{ selection } j\rangle$$

$$\langle\downarrow \text{ select } \rangle = \prod_{j=1}^{k}\langle\downarrow \text{ selection } j\rangle$$

Where Γ is a set of symbols, we use $\langle *\Gamma *\rangle$ to denote an arbitrary string which contains as a subsequence every string of length m over Γ (such as a string which simply runs through Γ m times in any order). As a notational convenience, we write $\langle *s...t*\rangle$ for $\langle *\Sigma_3 \cup \Sigma'[s,t]*\rangle$.

Recursively, we define $\langle\uparrow \alpha\rangle$ and $\langle\downarrow \alpha\rangle$ for $\alpha \in I$.

For $\alpha \in [n]^t$ and $l[\alpha] = u_i \in V$:

$$\langle\uparrow \alpha\rangle = p[\alpha]\left(\prod_{j=1}^{k} q[\alpha,j]u[\alpha,i,j]q'[\alpha,j]\right)p'[\alpha]$$

$$\langle\downarrow \alpha\rangle = p[\alpha]\left(\prod_{j=k}^{1} q[\alpha,j]u[\alpha,i,j]q'[\alpha,j]\right)p'[\alpha]$$

In general:

$$\begin{aligned}
\langle\uparrow\alpha\rangle &= p[\alpha]^{w(|\alpha|,t)}\prod_{i=1}^{n}\langle\uparrow\alpha.i\rangle p'[\alpha]^{w(|\alpha|,t)} \\
\langle\downarrow\alpha\rangle &= p[\alpha]^{w(|\alpha|,t)}\prod_{i=1}^{n}\langle\downarrow\alpha.i\rangle p'[\alpha]^{w(|\alpha|,t)} \qquad \text{for } \alpha \text{ even} \\
\langle\downarrow\alpha\rangle &= p[\alpha]^{w(|\alpha|,t)}\prod_{i=n}^{1}\langle\downarrow\alpha.i\rangle p'[\alpha]^{w(|\alpha|,t)} \qquad \text{for } \alpha \text{ odd}
\end{aligned}$$

The Reduction We may now describe the reduction. The instance of LCS-A to which we reduce consists of three sets of strings: the *Control Strings*, the *Quorum Strings* and the *Consistency Strings*.

The two *Control Strings* are

$$X_1 = \langle\uparrow \text{ select }\rangle\langle\uparrow\epsilon\rangle$$

$$X_2 = \langle\downarrow \text{ select }\rangle\langle\downarrow\epsilon\rangle$$

Let $\Delta = \{(r,s) : 1 \le r < s \le n\}$ ordered lexicographically. The $\binom{k}{2}$ *Quorum Strings* are, for $1 \le i < j \le n$

$$Y_{ij} = \left(\prod_{(r,s)\in\Delta}^{\uparrow\text{lex}} \langle *c[1]\cdots c[i]*\rangle v[r,i]\langle *c'[i]\cdots c[j]*\rangle v[s,j]\langle *c'[j]\cdots c'[k]*\rangle\right)\langle *\Sigma''*\rangle$$

The $2k$ *Consistency Strings* are, for $j = 1,\ldots,k$

$$Z_j = \prod_{r=1}^{n}\langle \text{ selection } j \text{ is variable } r \rangle$$

$$Z'_j = \prod_{r=n}^{1}\langle \text{ selection } j \text{ is variable } r \rangle$$

where

$$\begin{aligned}
\langle \text{ selection } j \text{ is variable } r \rangle &= \langle *c[1]\cdots c[j]*\rangle v[r,j]\langle *c'[j]\cdots c'[k]*\rangle \\
&\cdot \left(\prod_{\alpha\in[n]^t}^{\uparrow lex}\langle *q[\alpha,1]\cdots q[\alpha,j]*\rangle u[\alpha,r,j]\langle *q'[\alpha,j]\cdots q'[\alpha,k]*\rangle\right)
\end{aligned}$$

Proof of Correctness The following general ideas are useful to our arguments. To the expression E there naturally corresponds a Boolean tree circuit $C = C_E$. A truth assignment τ to the variables V of E may be considered as an input vector x_τ to the circuit C, with $C(x_\tau) = 1$ if and only if τ satisfies E. The circuit C maybe described:

(1) for each $\alpha \in I$, there is a gate g_α of C (of *rank* $|\alpha|$),

(2) g_α is an *and* gate if α is even, and an *or* gate if α is odd,

(3) the output gate of C is g_ϵ,

(4) for $|\alpha| < t$ the gate g_α takes input from the gates $g_{\alpha.i}$ for $i = 1, ..., n$,

(5) the inputs to C are in 1:1 correspondence with V, and

(6) for $|\alpha| = t$, the gate g_α takes the single input $u_i \in V$ such that $l[\alpha] = u_i$ in E.

A subcircuit C' of C is a *witnessing subcircuit* if it satisfies the conditions:

(1) $g_\epsilon \in C'$,

(2) for each even $\alpha \in I$, $|\alpha| < t$, if $g_\alpha \in C'$ then for all $i \in [n]$, $g_{\alpha.i} \in C'$, and

(3) for each odd $\alpha \in I$, if $g_\alpha \in C'$ then there is a unique $i \in I$ such that $g_{\alpha.i} \in C'$.

The following observations about witnessing subcircuits are useful.

Claim 1. $C(x) = 1$ if and only if there is a witnessing subcircuit C' of C such that $C'(x) = 1$ and each gate of C' evaluates to 1.

Claim 1 follows trivially from the monotonicity of C.

Claim 2. If C' is a witnessing subcircuit of C then the number of gates of rank r, for $r = 0, ..., t$, is given by the function

$$c(j) = n^{\lceil j/2 \rceil}$$

Claim 2 follows by an elementary induction, noting the special structure of C.

The following fact about the "weighting function" $w(j, t)$ will be useful.

Claim 3. For $0 \leq r \leq t-1$,

$$w(r,t) > \sum_{j=r+1}^{t} |X_1 \cap \Sigma''[j]|$$

Claim 3 is easily verified from the definitions.

Claim 4. In the Control Strings X_1 and X_2:

(1) Each symbol in Σ_3 occurs as a *block*, that is, the symbol occurs only in a substring consisting of some number of repetitions of the symbol.

(2) If $\alpha \prec \beta$ then all symbols with index β occur between the block of symbols $p[\alpha]^{w(|\alpha|,t)}$ and the block of symbols $p'[\alpha]^{w(|\alpha|,t)}$.

(3) If β is an index of a symbol occurring between the symbol blocks $p[\alpha]^{w(|\alpha|,t)}$ and $p'[\alpha]^{w(|\alpha|,t)}$ then $\alpha \preceq \beta$, with $\alpha \prec \beta$ properly if the symbol is in Σ_3.

Claim 4 is readily observed from the definition of $\langle \uparrow \alpha \rangle$ and $\langle \downarrow \alpha \rangle$.

In one direction, the argument for the correctness of the reduction is relatively easy. Given a satisfying weight k truth assignment $\tau : V \to \{0,1\}$ for E, we describe a common subsequence of length m in the following way. Let C' be a witnessing subcircuit of C for the input vector corresponding to τ. Let I' denote the set of indices of the logic gates of C'

$$I' = \{\alpha : g_\alpha \in C'\}$$

and suppose the variables set to 1 by τ are $v_{i_1}, ..., v_{i_k}$, with $v_{i_1} < v_{i_2} < \cdots < v_{i_k}$.

Let Γ denote the set of symbols

$$\begin{aligned} \Gamma \quad = \quad & \Sigma_1 \cup \{v[i_j, j] : 1 \leq j \leq k\} \cup \{p[\alpha], p'[\alpha] : \alpha \in I'\} \\ & \cup \{q[\alpha, j], q'[\alpha, j] : \alpha \in [n]^t \cap I',\ l[\alpha] = v_{i_j},\ 1 \leq j \leq k\} \\ & \cup \{u[\alpha, i_j, j] : \alpha \in [n]^t \cap I',\ l[\alpha] = v_{i_j},\ 1 \leq j \leq k\} \end{aligned}$$

Claim 5. The string $S = X_1 \cap \Gamma$ is a common subsequence of the Control and Consistency Strings of length m.

Proof of Claim 5. First note that $S = S_1 S_2$ where $S_1 \in (\Sigma_1 \cup \Sigma_2)^*$ and $S_2 \in (\Sigma'')^*$ and that similar factorizations hold for X_1 and X_2. An inspection of the definition of $\langle \uparrow \text{ select } \rangle$ and $\langle \downarrow \text{ select } \rangle$ shows that S_1 is a common subsequence of the first parts of X_1 and X_2, the main point being that between each pair of symbols $c[j]$ and $c'[j]$ there is just the single symbol $v[i_j, j]$ in S. Note also that the length of S_1 is $3k$.

Let $X_i' = X_i \cap \Sigma''$ for $i = 1, 2$. We argue that S_2 is a subsequence of X_1' and X_2' by inducting on the rank of symbols in S_2. Let $S_2[r]$ denote the subsequence of S_2 consisting of the symbols of rank r. By Claim 4, it is sufficient to establish that $S_2[r]$ is a subsequence of X_1' and X_2' for $r = 0, ..., t$. The base step of the induction, $r = 0$, is trivial. For the induction step, by Claim 4, it suffices to show that the subsequence of S consisting of symbols with index $\beta = \alpha.i$ (for some i) having rank $r+1$ is a subsequence of X_i' (for $i = 1, 2$) occurring between the symbol blocks $p[\alpha]^{w(|\alpha|,t)}$ and $p'[\alpha]^{w(|\alpha|,t)}$. If α is even then this follows from the fact that the blocks of the symbols $p[\alpha.i]$ and $p'[\alpha.i]$ occur in ascending order in both X_1' and X_2'. If α is odd then this follows trivially because there is only one relevant index $\alpha.i \in I'$. Note that in S there are precisely 3 symbols between each pair of symbol blocks $p[\alpha]^{w(|\alpha|,t)}$ and $p[\alpha]^{w(|\alpha|,t)}$ where $\alpha \in [n]^t \cap I'$, and that there are $n^{t/2}$ such pairs. ¿From this it is easy to verify that S has length m.

The above arguments establish that S is a subsequence of the Control Strings. By essentially the same inductive argument, the symbols $p[\alpha]$ must occur in S in lexicographically increasing order. Using this fact it is straightforward to verify that S is a subsequence of $\langle$ selection j is variable $i_j \rangle$ and thus S is a subsequence of Z_j and Z_j' for $j = 1, ..., k$. S_2 is trivially a subsequence of $\langle * \Sigma'' * \rangle$. For $p < q$, $v_{i_p} < v_{i_q}$, so S is a subsequence of Y_{pq}. □

To complete the proof of correctness for the reduction, we argue that if T is a common subsequence of the Control and Consistency Strings of length m then E is satisfied by a weight k truth assignment. In particular, we argue that T must correspond to a weight k input vector and a witnessing subcircuit of $C = C_E$ with respect to this vector.

Because the Control Strings can be factored in a similar way, we may factor T as $T = T_1 T_2$

with $T_1 \in (\Sigma_1 \cup \Sigma_2)^*$ and $T_2 \in (\Sigma'')^*$.

Claim 6. The length of T_1 is at most $3k$.

Claim 6 follows simply from the fact that for any fixed index j the symbols $v[i,j]$ occur in X_1 in increasing order with respect to i and they occur in X_2 in decreasing order with respect to i.

Say that an index $\alpha \in I$ is *represented* in T if both of the symbols $p[\alpha]$ and $p'[\alpha]$ occur in T. Say that an index $\alpha \in I$ is *forbidden* in T if for all indices β with $\alpha \preceq \beta$, no symbol with index β occurs in T. The following is an immediate consequence of the definition.

Claim 7. If α is forbidden in T, then for all $i \in [n]$, $\alpha.i$ is forbidden.

Claim 8. If $\alpha \in I$ is odd, then there is at most one $i \in [n]$ with $\alpha.i$ represented in T. Furthermore, if $\alpha.i$ is represented in T, then for all $j \neq i$, $\alpha.j$ is forbidden in T.

Proof of Claim 8. Suppose $i < j$ with $\alpha.i$ represented in T. By the definition of X_1 and X_2, all of the symbols with index $\alpha.i$ precede all of the symbols with index $\alpha.j$ in X_1, and all of the symbols with index $\alpha.i$ succeed all of the symbols with index $\alpha.j$ in X_2. Consequently no symbol with index $\alpha.j$ can occur in the common subsequence T. Furthermore, if β is an index with $\alpha.j \preceq beta$, then by Claim 4, all symbols with index β occur in X_1 and X_2 between blocks of symbols with index $\alpha.j$, so the above argument applies as well to symbols with these indices, so that $\alpha.j$ is forbidden. The case of $j < i$ is symmetric. □

Let $s(r)$ denote the number of indices $\alpha \in I$ of rank r that are represented in T.

Claim 9. For $r = 0, ..., t$

(1) $s(r) = c(r)$

(2) Every index of rank r is either represented or forbidden.

Proof of Claim 9. By induction on r. For $r = 0$, if either $p[\epsilon]$ or $p'[\epsilon]$ fails to occur in T, then necessarily $|T \cap \Sigma''[0]| \leq w(0,t)$, so T must contain at least $w(0,t)$ symbols of rank at least 1, a contradiction of Claim 3. This establishes both (1) and (2).

For the induction step, if $s(r+1) < c(r+1)$ then the induction hypothesis and the definition of m imply that T must contain more than $w(r+1,t)$ symbols of rank greater than $r+1$, which contradicts Claim 3. Suppose $s(r+1) > c(r+1)$.

Case 1: r is even. Then $c(r+1) = n \cdot c(r)$. By (1) of the induction hypothesis, there are precisely $s(r) = c(r)$ indices of rank r represented in T, and all other indices of rank r are forbidden. Since each represented index of rank r has only n extensions to an index of rank $r+1$, there must be some rank $r+1$ index $\alpha.i$ represented in T for which α is not represented in T. By (2) of the induction hypothesis, α is forbidden in T, a contradiction. Thus (1) must hold, and by the same argument, if α of rank r is represented then for all $i \in [n]$, $\alpha.i$ is represented. If α of rank r is forbidden in T, then by Claim 7, $\alpha.i$ is forbidden in T for all $i \in [n]$. This establishes (2).

Case 2: r is odd. Then $c(r+1) = c(r)$. By (1) of the induction hypothesis there are precisely $s(r) = c(r)$ indices of rank r represented in T, and all other indices of rank r are forbidden. There cannot be an index $\alpha.i$ represented in T with α not represented, as this would contradict (2) of the induction hypothesis. By the Pigeonhole Principle, there must be an index α represented in T and $i \neq j$ with both $\alpha.i$ and $\alpha.j$ represented in T, a contradiction of Claim 8. Thus (1) must hold, and by the same arguments we see that for each represented α of rank r there is a unique $i \in [n]$ with $\alpha.i$ represented. By Claim 8 we get (2). □

One can observe from the definition of $\langle\uparrow \alpha\rangle$ and $\langle\downarrow \alpha\rangle$ that there can be at most 3 symbols in $T \cap (\Sigma_4 \cup \Sigma_5)$ with a given index α of rank t, and that these must occur between $p[\alpha]$ and $p'[\alpha]$ and must occur in a substring of the form: $q[\alpha,j]u[\alpha,i,j]q'[\alpha,j]$. By this observation and Claims 6 and 9 we can conclude:

Claim 10. The length of T_1 is precisely $3k$ and the length of T_2 is precisely

$$3n^{t/2} + 2\sum_{j=0}^{t} c(j)w(j,t)$$

On the basis of Claim 10 we may associate to T a well-defined truth assignment τ to the variables of the expression E: $\tau(u_i) = 1$ if and only if for some j, $1 \leq j \leq k$, the symbol $v[i,j]$

occurs in T_1. By Claim 10, there are exactly k symbols of Σ_2 in T_1. However, we must argue that for $j < j'$, only one of $v[i,j]$ and $v[i,j']$ occurs in T_1, thus insuring that τ has weight k. To see this, note that T must be a subsequence of $Y_{jj'}$. Suppose $v[i,j]$ occurs in T (necessarily in T_1). The only symbols $v[i',j']$ occurring in $Y_{jj'}$ after $v[i,j]$ satisfy $i < i'$, by the definition of $Y_{jj'}$. Thus to T we may associate a truth assignment of weight k.

Claim 11. If $\alpha \in [n]^t$ is represented in T, with $u[\alpha,i,j]$ occurring between $p[\alpha]$ and $p'[\alpha]$, then $v[i,j]$ occurs in T_1 (and τ assigns u_i the value 1).

Proof of Claim 11. By Claim 10, there must be a symbol $v[p,j]$ in T_1. The symbol $u[\alpha,i,j]$ occurs only once in Z_j and in Z'_j. The symbols $v[p,j]$ preceding the occurrence of $u[\alpha,i,j]$ in Z_j satisfy $p \leq i$. The symbols $v[p,j]$ preceding the occurrence of $u[\alpha,i,j]$ in Z'_j satisfy $p \geq i$. Thus the only possibility is $v[i,j]$. □

Claim 12. The indices $\alpha \in I$ represented in T are those of a witnessing subcircuit C' of C that accepts the input vector x_τ corresponding to the truth assignment τ.

Proof of Claim 12. That the indices represented in T form a witnessing subcircuit C' of C follows from Claims 8 and 9. Since C' is monotone, it suffices to establish that all gates of rank t evaluate to 1 in order to conclude that $C'(x_\tau) = 1$. This follows from Claim 11, noting that if α of rank t is represented, then $u[\alpha,i,j]$ occurs between $p[\alpha]$ and $p'[\alpha]$ if and only if $l[\alpha] = u_i$, by the definition of $\langle \uparrow \alpha \rangle$ and Claim 4. □

By the correspondence between C and E, we conclude that τ is a weight k truth assignment for E, which establishes the validity of the first reduction.

The Second Reduction

Suppose we have sequences X_i, $1 \leq i \leq k$, over an alphabet of unrestricted size $\Sigma = \{v[1],...,v[s]\}$. We may assume, without loss of generality (by padding) that each sequence X_i has length n. Let m be a positive integer.

We describe how to compute from the above: (1) a set of sequences (Y_i), $1 \leq i \leq k$, and Z

over a new alphabet Γ of size $k+2$, and (2) a positive integer m', such that there is subsequence of length m' common to the sequences (Y_i) and the sequence Z if and only if there is a subsequence of length m common to the sequences (X_i).

The positive integer m' is described as follows. Let $l = n(s+2)+1$. Then $m' = (n+1)kl + m(s+2)$.

The alphabet Γ for the second reduction is described

$$\Gamma = \{a[i] : 1 \leq i \leq k\} \cup \{b, c\}$$

The following substring gadgets are useful.

$$T = \prod_{j=1}^{k} a[j]$$

$$T_i = \prod_{\substack{1 \leq j \leq k \\ i \neq j}} a[j]$$

$$U_i = T_i^{m'} a[i] T_i^{m'}$$

The target strings of the reduction are described

$$Z = \left(T^l b^s c b^s\right)^m T^l$$

and for $1 \leq i \leq k$

$$Y_i = \left(\prod_{\substack{j=1 \\ X_i[j] = v[r]}}^{n} U_i^l b^r c b^{s-r+1} \right) \cdot U_i^l$$

We will refer to the substring factors U_i^l of Y_i as *blocks* (thus each Y_i has $n+1$ of them). The substring factors between the blocks ($b^r c b^{s-r+1}$ for some r) we will term *zones*. It should be clear how each zone encodes a symbol of the unbounded alphabet Σ by placing the symbol c in an indexing position.

Proof of Correctness

Note that the symbol $a[i]$ occurs exactly $(n+1)l$ times in Y_i, l times in each of the $n+1$ blocks. For the remainder of the proof, let C denote a common subsequence of the set of strings $\{Z\} \cup \{Y_i : 1 \leq i \leq k\}$. The above observation implies that C contains at most $(n+1)kl$ symbols of type "a". Also note that each Y_i contains exactly $n(s+2)$ symbols of type "b" or "c", and consequently C contains at most $n(s+2)$ symbols of type "b" or "c".

Claim 13. If C is a common subsequence of length m' then for every i, $1 \leq i \leq k$, and for every possible way that C can occur as a subsequence of Y_i, C has nontrivial intersection with each block of Y_i.

Proof of Claim 13. If some block of Y_i is missed, then the symbol $a[i]$ occurs at most nl times in C, and C thus contains at most $(n+1)kl - l$ symbols of type "a". Since C can contain at most $n(s+2)$ symbols of type "b" or "c" this implies that $|C| \leq (n+1)kl - l + n(s+2) < (n+1)kl < m'$ (since $l > n(s+2)$), a contradiction. □

For the remainder of the argument, suppose C has length m', and in each of the sequences Z and Y_i fix attention on a particular C-subsequence.

Claim 14. For each i, $1 \leq i \leq k$, C has a nontrivial intersection with at most m zones of Y_i.

Proof of Claim 14. By Claim 13, C could not otherwise be a subsequence of Z, since the structure of Z limits the number of times C can alternate between symbols of type "a" and symbols of type "b" or "c". □

Claim 15. For each i, $1 \leq i \leq k$, if any symbol of a zone or block occurs in $C \cap Y_i$, then every symbol of the zone or block occurs in C.

Proof of Claim 15. By Claim 14, there can be at most $m(s+2)$ symbols of type "b" or "c" in the common subsequence C, and so by our initial observations there must be exactly this many symbols of type "b" or "c" in C. By Claim 13 these must occur in m zones of Y_i, and so each of these zones must be entirely contained in C. □

By the above arguments, if C is a common subsequence of length m' then it contains (entirely) m zones from each of the Y_i. By the construction of these zones, this yields a common subsequence of length m for the strings X_i over the large alphabet Σ.

Conversely, if D is a common subsequence of the strings (X_i) of length m, then we can describe the following common subsequence D' of length m' for the strings (Y_i) and Z. Let g_{ij}, $1 \leq i \leq k$ and $1 \leq j \leq m$ denote the jth *gap length* in X_i defined by fixing a subsequence D_i of X_i isomorphic to D, and letting g_{ij} be the unique positive integer such that $X_i[r] = D_i[j-1]$ and $X_i[r+g_{ij}] = D_i[j]$. (Let $g_{i,m+1}$ denote the "remaining" number of symbols in X_i in the natural way.)

$$D' = \prod_{j=1}^{m} \left[\left(\prod_{i=1}^{k} a[i]^{(g_{ij}+1)l} \right) \cdot b^{d(j)+1} c b^{s-d(j)} \right] \cdot \prod_{i=1}^{k} a[i]^{g_{i,m+1}}$$

where $d(j)$ is defined by the requirement

$$D[j] = v[d(j)] \in \Sigma$$

It is straightforward to verify that D' is a length m' common subsequence of the strings Z and Y_i $(1 \leq i \leq k)$.

4 Parameterized Problems in Computational Biology

The situation of the LCS problem in §3 is not unique; many problems in computational biology are known either to be NP-hard or to have only $O(n^k)$ algorithms. To solve such problems in practice, investigators must often settle for suboptimal solutions obtained by algorithms that are fast but are either approximate or solution-constrained [KS83, Pev92, Gus93]. Fixed-parameter algorithms are useful because they can provide exact solutions to the most commonly encountered (albeit smallest) instances of these problems. Moreover, even if we show that such algorithms probably do not exist by analyses like that given in §3, these same analyses establish the contribution that each parameter makes to a problem's complexity and thus suggest constraints that may make restricted versions of these problems practical.

In this section, we describe several problems from three areas of computational biology whose parameterized versions are of interest.

4.1 Multiple Sequence Alignment

Given a set of strings $X = \{x_1, \ldots, x_k\}$ on an alphabet Σ, an *alignment* of X is a set of strings $A = \{a1, \ldots, a_k\}$, $|a_1| = |a_2| = \ldots = |a_k| = n$, on augmented alphabet $\Gamma = \Sigma \cup \{\Delta\}$ such that each string a_i is a copy of x_i into which $n - |x_i|$ copies of special symbol Δ have been inserted [Pev92]. Symbol Δ is called an *indel* and represents the insertion or deletion of a particular symbol in one string relative to another. Let a_{ij} be the symbol in the j-th position of string a_i, and A_j, $1 \leq j \leq n$, be the k-vector $\{a_{1j}, \ldots, a_{kj}\}$ of symbols appearing in position j of the strings of A. Given a cost function $c : \Gamma^k \mapsto \mathcal{R}$ on A_j, *the cost of an alignment* A is the sum of the costs of all A_j. Two of the most commonly-used cost functions are constructed from a given $\Gamma \times \Gamma$ symmetric matrix M, where $M(x, y)$, $x, y \in \Gamma$ is the cost of converting symbol x to symbol y [Pev92]:

1. "Sum of Pairs" (SP) Function: $c_{SP}(A_j, M) = \sum_{1 \leq k < l \leq n} M(a_{kj}, a_{lj})$.

2. Tree Alignment (TA) Function: Define a *tree-alignment* (T, A) *of* X as an alignment A of X and a tree $T = (V, E)$ such that each vertex is labelled with a different string $x \in \Gamma^n$ and the strings of A are a subset of the vertex-labels of T. Let $l(v)$, $v \in V$, be the vertex-label string associated with vertex v in T. Then $c_{TA}(T, A_j, M) = \sum_{(x,y) \in E} M((l(x)_j, l(y)_j)$.

This yields the following variants of the general multiple sequence alignment problem:

Multiple Sequence TA Alignment (MSA-TA)

Input: Set of strings X, $|X| = k$, on alphabet Σ; a $\Gamma \times \Gamma$ symmetric matrix M; integer m.

Question: Is there a tree-alignment (T, A) of X such that $c_{TA}(T, A, M) \leq m$?

Multiple Sequence SP Alignment (MSA-SP)

Input: Set of strings X, $|X| = k$, on alphabet Σ; a $\Gamma \times \Gamma$ symmetric matrix M; integer m.

Question: Is there an alignment A of X such that $c_{SP}(A, M) \leq m$?

Though it is commonly stated that multiple alignment problems are NP-complete, the computational complexities of *MSA-TA* and *MSA-SP* are not known [Pev92, p. 1768]. The best known algorithms for *MSA-SP* and *MSA-TA* require $O(n^k)$ and exponential time, respectively (see [Gus93] and references). The k- and $|\Sigma|$-parameterized versions of these problems are useful for the same reasons as given in §3 for the LCS problem.

4.2 DNA Sequencing

Sequencing by Hybridization (SBH) is a technique for sequencing relatively short pieces of DNA which exploits the ability to detect the binding (hybridization) of a very short sequence of DNA (oligonucleotide) to its base-complementary region in a longer sequence (see [PLKBFM91] and references). In SBH methods, one hybridizes all possible oligonucleotides of a fixed length k against a given sequence, and then uses the pairwise overlaps of the set X of detected hybridizations to reconstruct the original sequence. For instance, the sequence ATCCGC can be reconstructed from set $X = \{$ATC,TCC,CCG,CGC$\}$.

Given an alphabet $\Sigma_{DNA} = \{A, C, T, G\}$ and a set $X \subseteq \Sigma_{DNA}^k$, a *proper overlapping s of X* is a string s such that every $x \in X$ appears at least once as a substring of s, and s is composed by successive overlaps of length $k-1$ of pairs of strings drawn from X. Polynomial time reconstruction algorithms are known when the number of occurrences of each $x \in X$ is known and there is a unique proper overlapping of X [Pev89]. The following problems correspond to the frequently-occurring instances in which one or both of these assumptions is violated:

Shortest SBH Reconstruction (SBH)

Input: Integer k; set $X \subseteq \Sigma_{DNA}^k$; integer l.

Question: Is there a proper overlapping s of X of length at most l?

Shortest SBH Reconstruction with Additions (SBH-ADD)

Input: Integer k; set $X \subseteq \Sigma_{DNA}^k$; integers l, m.

Question: Does there exist a set $Y \subseteq \Sigma_{DNA}^k$, $|Y| \leq m$, such that there is a proper overlapping s of

$(X \cup Y)$ of length at most l?

Shortest SBH Reconstruction with Deletions (SBH-DEL)

Input: Integer k; set $X \subseteq \Sigma_{DNA}^k$; integers l, m.

Question: Does there exist a set $Y \subseteq X$, $|Y| \leq m$, such that there is a proper overlapping s of $(X - Y)$ of length at most l?

The computational complexities of these problems are not known. The k-parameterized versions of each of these problems are of interest because it is unlikely that hybridization tests can be carried out quickly and reliably for complete oligonucleotide sets of length greater than 12 using current technology [PLKBFM91]. Even within this range of k, errors at various stages in processing can result in oligonucleotides being added to or deleted from X [SME92]. Hypotheses of such errors should be suggested only when necessary, and should be added incrementally so that users can explore the space of possible reconstructions that these errors allow. The m-parameterized versions of problems *SBH-ADD* and *SBH-DEL* are useful for adding hypotheses of error in this manner.

A more general version of the SBH reconstruction problem is: given a set of fragments of a sequence and a measure of overlap between each pair of sequences in this set, reconstruct the order of these fragments in the original sequence. This problem has occurred at several levels (proteins, gene sequences, chromosome sequences) in DNA physical mapping over the last fifty years [JDD82, p. 259]. Let the fragments correspond to the vertices of a graph G, and let each overlap be represented by an edge between the corresponding pair of vertices in G. If the overlap data is error-free and complete, then G is an interval graph, and there are polynomial-time algorithms for reconstructing the original fragment order (see [Gol80] and references). However, for a variety of reasons [MCZSL87, pp. 205-208], there may be errors in the data. Hence, the following problems may be useful.

Graph Intervalization by Additions (GI-ADD)

Input: Graph $G = (V, E)$; integer k.

Question: Is there an interval graph $G' = (V, E')$ such that $E \subseteq E'$ and $|E' - E| \leq k$?

Graph Intervalization by Deletions (GI-DEL)

Input: Graph $G = (V, E)$; integer k.

Question: Is there an interval graph $G' = (V, E')$ such that $E' \subseteq E$ and $|E - E'| \leq k$?

Graph Intervalization by Deletions and Additions (GI-DEL/ADD)

Input: Graph $G = (V, E)$; integers k, l.

Question: Is there an interval graph $G' = (V, E')$ such that $E' = (E - E'') \cup E'''$, $|E''| \leq k$, and $|E'''| \leq l$?

Problems *GI-ADD* and *GI-DEL* are NP-complete by [GJ79, Problem GT35] and [GCKS93, Theorem 3.1], respectively; problem *GI-DEL/ADD* is NP-complete by trivial reductions from either of these problems. As with the SBH problems above, the k- and l-parameterized versions of these problems are useful for the gradual addition of hypotheses of error.

4.3 3-D Biopolymer Folding

All known approaches to determining the 3-D structure of biopolymers from their sequences seem to require exponential time [Ree88] or are NP-hard [NM91]. One heuristic for solving these problems is to break the given sequence into smaller pieces that can be folded in reasonable time by existing algorithms. Many proteins seem to be composed of short (40-150 unit) regions that fold independently of other regions [Ree88, p. 263]. The theory of exon shuffling proposes that all proteins are concatenations of sets of these regions which are drawn from an ancestral dictionary containing several thousand such regions [DG91, Pat91]. The problem defined below involves deriving such an underlying dictionary.

Given a set of strings $D \subseteq \Sigma^*$ and a string $s \in \Sigma^*$, define a *factorization of s relative to D* as a string $d = d_1 d_2 \ldots d_m$ such that d_1, d_2, ..., $d_m \in D$ and $d = s$. Define the *length of a factorization* d as m. Note that d may contain repetitions of elements in D, and that there may be several factorizations of a given s relative to D.

Dictionary Generation (DICT-GEN)

Input: Set of strings $S \subseteq \Sigma^*$; integers k, l.

Question: Is there a set of strings $D \subseteq \Sigma^*$ such that $|D| \leq k$ and for each $s \in S$, there is a factorization of s relative to D of length at most l?

Though there are efficient algorithms for searching for similar regions in a given set of sequences or finding all occurrences of regions within a given dictionary in those sequences [Aho90, PM92], the computational complexity of this problem is not known. Parameters k and l interact in this problem to create the smallest possible dictionary composed of the largest substrings. These substrings are the weakest, and hence logically most desirable, hypotheses of structure that can be made within the exon shuffling theory in the absence of more further information about the underlying dictionary. The l-parameterized version of problem *DICT-GEN* may be useful in exploring the space of possible dictionaries for a given set of strings.

References

[Aho90] A. V. Aho, "Algorithms for Finding Patterns in Strings". In J. van Leeuwen (ed.) *Handbook of Theoretical Computer Science*, pp. 257–300, Elsevier Science Publishers B. V., Amsterdam, 1990.

[BFH93] H. L. Bodlaender, M. R. Fellows, and M. T. Hallett, "The Parameterized Complexity of DNA Mapping, Perfect Phylogeny and Other Problems of 'Bounded Width' (Extended Abstract)", to appear, *STOC 26*.

[BFW92] H. L. Bodlaender, M. R. Fellows, and T. J. Warnow, "Two Strikes Against Perfect Phylogeny," Utrecht University, Technical Report RUU-CS-92-08, Department of Computer Science, February, 1992.

[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.

[DG91] R. L. Dorit and W. Gilbert, 'The limited universe of exons", *Current Opinions in Structural Biology*, 1, 973–977, 1991.

[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.

[DM93] 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.

[FHW93] M. R. Fellows, M. T. Hallett, and H. T. Wareham, DNA Physical Mapping: Three Ways Difficult". In *ESA'93: Proceedings*, Springer-Verlag.

[GJ79] 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.

[Gol80] M. C. Golumbic, *Algorithmic Graph Theory and Perfect Graphs*, Academic Press, New York, 1980.

[GCKS93] P. W. Goldberg, M. C. Golumbic, H. Kaplan, and R. Shamir, "Three Strikes Against Physical Mapping of DNA (extended abstract)", manuscript.

[Gus93] D. Gusfield, "Efficient Methods for Multiple Sequence Alignment with Guaranteed Error Bounds", *Bulletin of Mathematical Biology*, 55(1), 141–154, 1993.

[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.

[JDD82] J. R. Jungck, G. Dick, and A. G. Dick, "Computer-assisted sequencing, interval graphs, and molecular evolution", *BioSystems*, 15, 259–272, 1982.

[KS83] J. B. Kruskal and D. Sankoff, "An Anthology of Algorithms and Concepts for Sequence Comparison". In D. Sankoff and J. B. Kruskal (eds.) *Time Warps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison*, pp. 265–310, Addison-Wesley, Reading, MA, 1983.

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

[MCZSL87] F. Michiels, A. G. Craig, G. Zehetner, G. P. Smith, and H. and Lehrach, "Molecular approaches to genome analysis: a strategy for the construction of ordered overlapping clone libraries," *Computer Applications in the Biosciences*, 3(3), 203–210, 1987.

[NM91] T. J. Ngo and J. Marks, "Computational Complexity of a Problem in Molecular Structure Prediction", Technical Report TR-17-91, Center for Research in Computing Technology, Aiken Computation Laboratory, Harvard University, 1991.

[Pat91] L. Patthy, "Exons - Original Building Blocks of Proteins?", *BioEssays*, 13(4), 187–192, 1991.

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

[Pev89] P. A. Pevzner "l-Tuple DNA Sequencing: Computer Analysis," *Journal of Biomolecular Structure & Dynamics*, 7(1), 63–73, 1989.

[Pev92] P. A. Pevzner, "Multiple Alignment, Communication Cost, and Graph Matching", *SIAM*

Journal on Applied Mathematics, 52(6), 1763-1779, 1992.

[PLKBFM91] P. A. Pevzner, Y. P. Lysov, K. R. Khrapko, A. V. Belyavsky, V. L. Florentiev, and A. D. Mirzabejov, "Improved Chips for Sequencing by Hybridization", *Journal of Biomolecular Structure & Dynamics*, 9(2), 399–410, 1991.

[Ree88] G. N. Reeke Jr., "Protein Folding: Computational Approaches to an Exponential-Time Problem", *Annual Reviews of Computer Science*, 3, 59–84, 1988.

[SME92] E. M. Southern, U. Maskos, and J. K. Elder "Analyzing and Comparing Nucleic Acid Sequences by Hybridization to Arrays of Oligonucleotides: Evaluation Using Experimental Models", *Genomics*, 13, 1008–1017, 1992.