Parameterized Complexity Analysis in Computational Biology

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

June 9, 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 a new result for 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 Sequence Alignment; Consensus Sequence; 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, Rod.Downey@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

1 Introduction

With the recent availability of large amounts of molecular sequence data, the number of opportunities for applying various types of sequence-comparison analyses, e.g., multiple alignment, consensus discovery, have increased dramatically. However, the number of sequences that can be examined at one time is often limited to less than six by the $O(n^k)$ time requirements of the best known algorithms for these analyses, where k is the number of sequences and n is the maximum number of symbols in any of the given sequences (Kruskal and Sankoff, 1983; Carrillo and Lipman, 1988; Timkovskii, 1990; Irving and Fraser, 1992). These requirements seem to be inherent in the dynamic programming paradigm in which many of these algorithms have been derived (see Pearson and Miller, 1992, and references).

Within the last decade, new algorithmic techniques derived from the work of Robertson and Seymour on graph minors (see Fellows, 1989, and references) have allowed some such problems to be solved efficiently, in particular those whose instances encountered in practice have small values for one of their parameters. For example, 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.

CUTWIDTH: Is there a linear ordering of V with cutwidth k, i.e., is there a 1:1 function $f : V \rightarrow \{1, 2, \ldots, |V|\}$ such that all i, $1 < i < |V|$, $|\{\{u, v\} \in E : f(u) \leq i < f(v)\}| \leq k$?

BANDWIDTH: Is there a linear ordering of V with bandwidth k, i.e., is there a 1:1 function $f : V \rightarrow \{1, 2, \ldots, |V|\}$ such that all $\{u, v\} \in E$, $|f(u) - f(v)| \leq k$?

Both of these problems have applications in VLSI circuit design (Fellows and Langston, 1992), both are NP-complete (Garey and Johnson, 1979, problems GT44 and GT40), and both have $O(|V|^k)$ dynamic programming algorithms. Yet, despite their superficial similarity, the first is solvable in linear time for fixed values of k using the Robertson-Seymour techniques while the second has resisted all such attacks.

The above example is not an isolated phenomena; there does not seem to be any correlation between the general, e.g., NP/PSPACE-hard, complexity of a problem and whether or not it will be fixed-parameter tractable. The theory of parameterized computational complexity introduced in Downey and Fellows (1992) was designed to address this natural and important qualitative complexity distinction. For example, within this theory, CUTWIDTH is known to be in class FPT (see §2), while BANDWIDTH is hard for all classes $W[t]$, $t \geq 1$ (see §2), and hence not fixed-parameter tractable unless such well-known problems as CLIQUE or WEIGHTED BINARY INTEGER PROGRAMMING are also fixed parameter-tractable and certain mathematical conjectures are proved false (Downey and Fellows, 1993; Cai *et al.*, 1994).

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 (Bodlaender *et al.*, 1992; Fellows *et al.*, 1993; Kaplan and Shamir, 1993; Bodlaender *et al.*, 1994a; Bodlaender *et al.*, 1994b, Kaplan *et al.*, 1994). 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 apply this theory to the LONGEST COMMON SUBSEQUENCE (LCS) problem when the the fixed parameter is the size of the alphabet as well as the number of given strings. 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

Theories of computational complexity can be used to show that feasible algorithms may not exist for particular computational problems by virtue of the following four components:

1. An appropriate universe U of computational problems;
2. A class F, $F \subseteq U$, of problems that have feasible algorithms;
3. A reducibility $\propto$ between problems in U that preserves feasibility, i.e., for $x, y \in U$, if $x \propto y$ and $y \in F$ then $x \in F$; and
4. A class C, $C \subseteq U$, of problems for which it is either known or strongly conjectured that for every $x \in C$, $x \notin F$.

Within such a theory, a problem $x \in U$ is *C-hard* if for all $y \in C$, $y \propto x$; if $x \in C$ as well, then x is *C-complete*. The importance of C-hardness/completeness is that if a problem x is C-hard/complete then x does not have a feasible algorithm (modulo the strength of the conjecture $(\forall x \in C)\ x \notin F$).

The most familiar theory of computational complexity used in this fashion is that for NP-completeness, in which components (1) – (4) above are the universe of decision problems, the class P, polynomial-time many-one reducibility, and the class NP-complete (Garey and Johnson, 1979). In a similar manner, one can also sketch the framework of parameterized complexity theory as follows (for greater detail, see Downey and Fellows, 1992).

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 $f : N \mapsto N$ is an arbitrary function and α is a constant independent of k. 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 $f, g : N \mapsto N$ 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; and (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.

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 $L_F = \{(C, k) : C$ 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. By definition, these classes form the following hierarchy.

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

It is conjectured that all inclusions in this hierarchy are proper (Downey and Fellows, 1993; Cai *et al.*, 1994). Known results within this hierarchy include:

- Gate matrix layout, Vertex cover, Steiner tree in graphs (in FPT),
- Clique, Short nondeterministic Turing machine computation, Vapnik-Chervonenkis dimension ($W[1]$-complete),
- Dominating set, Set cover, Weighted binary integer programming ($W[2]$-complete),
- Bandwidth, DNA physical mapping, Perfect phylogeny ($W[t]$-hard for $t \geq 1$), and
- Compact Turing machine computation, Minimum axiom set, and k-Based tiling ($W[P]$-complete).

Over one hundred such results are currently known and listed on-line (Hallett and Wareham, 1994) (see directory **pub/W_hierarchy** via anonymous ftp to **csr.uvic.ca**). In this framework, no $W[x]$-complete problem is fixed parameter tractable unless all problems in $W[x]$ are fixed-parameter tractable. Hence, modulo the results in Downey and Fellows (1993) and Cai *et al.* (1994), a W-hardness result for a problem suggests that $O(n^k)$ time algorithms might indeed be the best that we can do for that problem, cf. §4.

3 The Parameterized Complexity of the Longest Common Subsequence Problem

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 Irving and Fraser, 1992, and references). The k-unrestricted LCS problem is NP-complete even if the alphabet is of size two (Maier, 1978; see also Timkovskii, 1990), and the best known algorithms require $O(n^k)$ time and space (Irving and Fraser, 1992). Our interest in this problem comes from the fact that it is a special case of the problems of consensus subsequence discovery and multiple sequence alignment under arbitrary alignment evaluation functions (Pevzner, 1992; Day and McMorris, 1993; Kececioglu, 1993; Bodlaender *et al.*, 1994a). Thus complexity lower bounds on the LCS problem imply complexity lower bounds for these more complex and realistically formulated problems in biological computing.

Consider the complexity of the following parameterized versions of LCS.

LONGEST COMMON SUBSEQUENCE
Instance: A set of k strings $X_1, ..., X_k$ over an alphabet Σ, and a positive integer m.
Parameter 1 (LCS-1): k
Parameter 2 (LCS-2): m
Parameter 3 (LCS-3): k, m
Parameter 4 (LCS-4): $k, |\Sigma|$
Question: Is there a string $X \in \Sigma^*$ of length at least m that is a subsequence of X_i for $i = 1, ..., k$?

Let LCS-5 denote LCS-1 when the size of the alphabet Σ is a fixed constant. The parameterized complexities of these problems and several of their variants are shown in Table 1. Note that many of these problems become fixed-parameter tractable when m and $|\Sigma|$ are fixed in some fashion (this is by the trivial algorithm that generates all $|\Sigma|^m$ possible subsequence strings and checks them against each X_i). Our concern in this section is with LCS-4 and LCS-5.

Table 1: The Fixed-Parameter Complexity of the LCS Problem

	Alphabet Size $\lvert\Sigma\rvert$		
Parameter	Unbounded	Parameter	Constant
k	LCS-1 W[t]-hard for $t \geq 1$ [BDFW94,BFH94]	LCS-4 W[t]-hard for $t \geq 1$ (below)	LCS-5 ?
m	LCS-2 W[2]-hard [BDFW94]	FPT	FPT
k, m	LCS-3 W[1]-complete [BDFW94]	FPT	FPT

All of the k-parameterized versions of LCS are relevant because conventional $O(n^k)$ time algorithms can only handle instances of up to six strings, while an FPT algorithm might be able to handle upwards of 20 strings. The most compelling of these problems is LCS-5, 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-4 can be viewed as a kind of approximation to LCS-5. Our failure to find a hardness result for LCS-5 invites hope that it could be fixed-parameter tractable.

Theorem. LCS-4 is hard for $W[t]$ for all t.

Proof. The proof consists of a reduction from LCS-1 to LCS-4.

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 this reduction is described

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

The following substring gadgets are useful. Product notation in the description of these gadgets refers to string concatenation. Where s is a symbol, the notation s^w denotes the symbol s repeated w times.

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

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

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

The target strings of the reduction are

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

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*. Note that each Y_i has $n+1$ blocks. 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 1. 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 1. 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 2. For each i, $1 \leq i \leq k$, C has a nontrivial intersection with at most m zones of Y_i.

Proof of Claim 2. By Claim 1, 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 3. 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 3. By Claim 2, 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 1 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 $(1 \leq i \leq k)$, which completes the proof. □

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 (Kruskal and Sankoff, 1983; Pevzner, 1992; Gusfield, 1993; Wareham, 1993; Jiang and Li, 1994b). 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

As noted in §2, a longest common subsequence of a given set S of strings is not only a measure of agreement (or *consensus*) among these strings, but is also a guide for showing how parts of these strings relate to one another (*alignment*). In this section, we will consider more sophisticated types of 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 $A = \{a_1, \ldots, a_k\}$, $|a_1| = |a_2| = \ldots = |a_k| = n$, 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 (Pevzner, 1992; Kececioglu, 1993). 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. If arbitrary cost functions are allowed then the problem of finding the minimal cost alignment of a set of strings is NP-hard, because the LCS problem can be solved using a particular cost function (Pevzner, 1992; Kececioglu, 1993). 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.

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

2. Tree Alignment (TA) Function: Define a *tree-alignment T of X* as a tree $T = (V, E)$ such that each vertex is labelled with a different string from Γ^p, $p \geq \max_{x_i \in X} |x_i|$, and each string in X is a subsequence of a distinct vertex-label string in V. Let $l(v)$, $v \in V$, be the vertex-label string associated with vertex v in T, and $l(v_j)$ be the j-th character in that vertex-label string. Note that implicit in T is an alignment of the strings in X; denote this alignment over all vertex-label strings in T as T_A, and let T_{Aj} denote the j-th column of T_A. Then $c_{TA}(T_{Aj}, M) = \sum_{(x,y) \in E} M(l(x_j), l(y_j))$.

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

Multiple sequence SP alignment (MSA-SP)
Instance: A set X of k strings on alphabet Σ, a $\Gamma \times \Gamma$ symmetric matrix M, and an integer m.
Question: Is there an alignment A of X such that $c_{SP}(A, M) \leq m$?

Multiple sequence TA alignment (MSA-TA)
Instance: A set X of k strings on alphabet Σ, a $\Gamma \times \Gamma$ symmetric matrix M, and an integer m.
Question: Is there a tree-alignment T of X such that $c_{TA}(T, M) \leq m$?

Multiple sequence TA alignment with given tree (MSA-TAG)
Instance: A set X of k strings on alphabet Σ, a partial tree-alignment $T = (V, E)$ such that each string in X is a subsequence of a distinct vertex-label string in $V' \subseteq V$ and the vertices in $V - V'$ are unlabelled, a $\Gamma \times \Gamma$ symmetric matrix M, and an integer m.
Question: Is there an assignment of vertex-label strings to the unlabelled vertices of T, i.e., $V - V'$, such that $c_{TA}(T, M) \leq m$?

Problems MSA-SP and MSA-TA are NP-hard by Wang and Jiang (1994) and Sweedyk and Warnow (1994), respectively, and MSA-TAG is NP-complete when the given tree is binary and MAX SNP-hard when the given tree is a star (Wang and Jiang, 1994). The best known algorithms for MSA-SP require $O(n^k)$ time, and those for MSA-TA and MSA-TAG require exponential time (see Wang and Jiang, 1994, 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; it might also be interesting to investigate parameterizations of properties of the given trees in MSA-TAG, e.g., number of Steiner vertices, maximum path length between any pair of leaves.

One approach to sidestepping the complexity of k-sequence alignments is to compute all $\binom{k}{2}$ pairwise alignments, select a subset of these alignments, and merge these selected alignments into a multiple alignment (see Gotoh, 1993, Kececioglu, 1993, and references). Though such methods require only time polynomial in n and k, it is not hard to construct examples where the produced alignment agrees only with the subset of selected alignments (Kececioglu, 1993).

To address the question of finding a multiple alignment in this scheme that is as close as possible to all computed pairwise alignments, Kececioglu (1991, 1993) defined the following problem. Let the *pairwise alignment graph* $(V, E, \prec)$ of a set S of sequences be the graph whose vertices and edges correspond to characters in the sequences of S and pairwise alignments of sequences in S, respectively, and relation $v \prec w$ holds if and only if character v immediately precedes character w in the appropriate sequence in S. Given $X, Y \subseteq V$, let $X \prec^* Y$ if and only if $(\exists x \in X)(\exists y \in Y)$ such that $x \prec y$. A *trace* of an alignment graph $(V, E, \prec)$ is a subset $T \subseteq E$ such that $\prec^*$ on the connected components of T is acyclic. Note that a trace of maximum cardinality agrees with as many of the alignment-induced character-matches as possible.

Maximum weight trace (MWT)
Instance: An alignment graph $(V, E, \prec)$ with an edge-weight function w, an integer l.
Question: Is there a trace $T \subseteq E$ such that $\sum_{e \in T} w(e) \geq l$?

This problem is NP-complete by (Kececioglu, 1993) even when w assigns unit weight to each edge in E. This problem can be solved in $O(n^k)$ time by dynamic programming (Kececioglu, 1991), and has a worst-case exponential time branch-and-bound algorithm (Kececioglu, 1993). The k- and $|\Sigma|$-parameterized versions of this problem are of interest, especially as problem MSA-SP above under symbol-independent insertion and deletion costs is a special case of MWT (Kececioglu, 1993).

4.2 Sequence Reconstruction

Current technology allows only relatively short regions of DNA or protein to be sequenced; hence, the base-sequences of longer regions must be determined by breaking such regions into fragments that can be sequenced and then reconstructing the region from these fragments. In much the same way as the LCS problem underlies various versions of multiple sequence alignment and consensus, the following problem underlies sequence reconstruction:

SHORTEST COMMON SUPERSTRING (SCS)
Instance: A set of k strings $X_1, \ldots, X_k$ over an alphabet Σ, and an integer m.
Question: Is there a string $X \in \Sigma^*$ of length at most m such that each X_i is a substring of X for $i = 1, \ldots, k$?

This problem is NP-complete when $|\Sigma| \geq 2$ by Gallant *et al.* (1980). As the number of fragments may be very large, k-parameterizations of SCS are of little interest; however, other parameters, e.g., maximum number of fragments overlapping on the superstring, are still relevant. There are a number of variants of this problem that may also yield relevant parameterizations (Timkovskii, 1990; Kececioglu, 1991; Jiang and Li, 1993; Jiang and Li, 1994a; Middendorf, 1994).

A restricted case of the SCS problem occurs in Sequencing by Hybridization (SBH). Sequencing by Hybridization 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 (Pevzner *et al.*, 1991; Pevzner and Lipshutz, 1994). 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 (Pevzner, 1989; Pevzner and Lipshutz, 1994). The following problems correspond to the frequently-occurring instances in which one or both of these assumptions is violated:

SHORTEST SBH RECONSTRUCTION (SBH)
Instance: An integer k, a set $X \subseteq \Sigma_{DNA}^k$, and an integer l.
Question: Is there a proper overlapping s of X of length at most l?

SHORTEST SBH RECONSTRUCTION WITH ADDITIONS (SBH-ADD)
Instance: An integer k, a set $X \subseteq \Sigma_{DNA}^k$, and 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)
Instance: An integer k, a set $X \subseteq \Sigma_{DNA}^k$, and 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 (Pevzner *et al.*, 1991; Pevzner and Lipshutz, 1994). Even within this range of k, errors at various stages in processing can result in oligonucleotides being added to or deleted from X (Southern *et al.*, 1992; Pevzner and Lipshutz, 1994). 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 sequence reconstruction problem is: given a set of fragments of a sequence and a measure of sequence overlap between each pair of fragments 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 (Jungck *et al.*, 1982, 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 Golumbic (1980) and references). However, for a variety of reasons (Michiels *et al.*, 1987, pp. 205–208), there may be errors in the data. Hence, the following problems may be useful.

GRAPH INTERVALIZATION BY ADDITIONS (GI-ADD)
Instance: A graph $G = (V, E)$ and an 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)
Instance: A graph $G = (V, E)$ and an 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)
Instance: A graph $G = (V, E)$ and an integer k.
Question: Is there an interval graph $G' = (V, E')$ such that the symmetric difference of E and $E' \leq k$?

Problems GI-ADD and GI-DEL are NP-complete by Garey and Johnson (1979) (Problem GT35) and Goldberg *et al.* (1993), respectively; the complexity of problem GI-DEL/ADD is unknown. As with the SBH problems above, the k-parameterized versions of these problems are useful for the gradual addition of hypotheses of error. Various NP-completeness and W-hardness results have recently been derived for versions of these problems that have been restricted to fragments of unit length, fragments that are not allowed to be properly included inside each other, and solution interval graphs of bounded clique size (Goldberg *et al.*, 1993; Kaplan and Shamir, 1993; Golumbic *et al.*, 1994; Kaplan *et al.*, 1994). Note that these problems can also be stated on colored graphs, yielding alternate (and more biologically useful) versions of the problem INTERVALIZING COLORED GRAPHS studied in Fellows *et al.* (1993) and Bodlaender *et al.* (1994b).

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 (Reeke, 1988). A potentially more tractable approximate model, the lattice model, has been studied extensively over the last seven years, both via statistical mechanics and computational simulations (Chan and Dill, 1991, 1993; Frauenfelder and Wolynes, 1994). In such models, polymers are folded such that their basic units, e.g., nucleotides, amino acids, are restricted to intersection points on a 2-D grid or 3-D cubic lattice. Given a sequence over an alphabet Σ and a $|\Sigma| \times |\Sigma|$ base-interaction strength matrix M, the optimal (and presumed natural) fold of the sequence is that configuration of the sequence on the lattice such that the sum of interaction strengths between all positions in the sequence is maximized. A particularly simple version of this model studied by Dill and his colleagues (see Chan and Dill, 1993, and references) considers only those interactions between bases that are adjacent on the lattice but not adjacent in the sequence. For a base s_i in a sequence s and a fold f of s on a lattice, let $A(s_i, f)$ be the set of bases in s that can interact with s_i.

d-LATTICE PROTEIN FOLDING (dLPF)
Instance: A sequence s on alphabet Σ, a $|\Sigma| \times |\Sigma|$ matrix M, an integer l.
Question: Is there a fold f of s on the d-dimensional lattice such that $\sum_{i=1}^{|s|} \sum_{x \in A(s_i,f)} M(x, s_i) \geq l$?

INVERSE d-LATTICE PROTEIN FOLDING (IdLPF)
Instance: A fold f of length n on a d-dimensional lattice, an alphabet Σ, a $|\Sigma| \times |\Sigma|$ matrix M.
Question: Is there a sequence $s \in \Sigma^n$ such that f is an optimal-cost fold for s under M?

The first problem is useful in determining the fold of a particular sequence, while the second is useful for designing sequences that fold to a particular structure, i.e., rational drug design (Martin, 1991). Though certain polynomial-time heuristics have been derived for these problems (Yue and Dill, 1992, 1993), the exact complexity of these problems is not known. The d and $|\Sigma|$-parameterized versions of these problems are of obvious interest for $d = 2, 3$ and $2 \leq |\Sigma| \leq 20$. Though the solutions given by these models are typically not good approximations to actual folds, they are very useful in testing hypotheses about the mechanisms by which proteins fold as well as the properties of actual folds (see Chan and Dill, 1993, and references). Though relevant parameterizations are much more difficult to formulate, it may also be interesting to examine versions of this model that allow non-local interactions (Fraenkel, 1993; Unger and Moult, 1993) as well as more realistic models (Ngo and Marks, 1992).

One heuristic for solving such folding 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 (Reeke, 1988, 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 (Dorit and Gilbert, 1991; Patthy, 1991). 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 $d_1, d_2, \ldots, d_m \in D$ 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)
Instance: A set of strings $S \subseteq \Sigma^*$ and 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 listed in a given dictionary in those sequences (Aho, 1990; Pearson and Miller, 1992), the 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 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.

Acknowledgements

H. T. Wareham would like to thank W. Day, K. Dill, T. Jiang, J. Kececioglu, M. Li, M. Middendorf, P. Pevzner, R. Shamir, and P. Wolynes both for pleasant discussions and for their generosity in sharing reprints and manuscripts.

References

Aho, A. V. (1990) Algorithms for finding patterns in strings. In van Leeuwen, J. (ed). *Handbook of Theoretical Computer Science Volume A: Algorithms and Complexity.* Elsevier Science Publishers B. V., Amsterdam, pp. 257–300.

Bodlaender, H. L., Downey, R. G., Fellows, M. R. and Wareham, H. T. (1994a) The parameterized complexity of sequence alignment and consensus. In Crochemore, M. and Gusfield, D. (eds). *Proceedings of the Fifth Annual Symposium on Combinatorial Pattern Matching (CPM).* Lecture Notes in Computer Science no. 807. Springer-Verlag, Berlin, pp. 15–30. To appear, *Theor. Comput. Sci.*

Bodlaender, H. L., Fellows, M. R. and Hallett, M. T. (1994b) The parameterized complexity of DNA Mapping, Perfect Phylogeny and other problems of 'bounded width' (extended abstract). In *Proceedings of the 26th Annual ACM Symposium on the Theory of Computing (STOC).* ACM Press, pp. 449–458.

Bodlaender, H. L., Fellows, M. R. and Warnow, T. J. (1992) Two strikes against Perfect Phylogeny. Technical report RUU-CS-92-08, Department of Computer Science, Utrecht University.

Cai, L., Chen, J., Downey, R. G. and Fellows, M. R. (1994) On the structure of parameterized problems in NP. In *STACS'94: 11th Annual Symposium on the Theoretical Aspects of Computer Science.* Lecture Notes in Computer Science. Springer Verlag, Berlin, pp. 509–520.

Carrillo, H. and Lipman, D. (1988) The multiple sequence alignment problem in biology. *SIAM J. Appl. Math.*, **48(5)**, 1073–1083.

Chan, H. S. and Dill, K. A. (1991) Polymer principles in protein structure and stability. *Annual Review of Biophysics and Biophysical Chemistry*, **20**, 447–490.

Chan, H. S. and Dill, K. A. (1993) The protein folding problem. *Physics Today*, **Feb.**, 24–32.

Day, W. H. E. and McMorris, F. R. (1993) The computation of consensus patterns in DNA sequences. *Mathematical and Computer Modelling*, **17(10)**, 49–52.

Dorit, R. L. and Gilbert, W. (1991) The limited universe of exons. *Current Opinions in Structural Biology*, **1**, 973–977.

Downey, R. G. and Fellows, M. R. (1992) Fixed-parameter intractability (extended abstract). In *Proceedings of the Seventh Annual Conference on Structure in Complexity Theory.* IEEE Computer Society Press, Los Alamitos, CA, pp. 36–49.

Downey, R. G. and Fellows, M. R. (1993) Fixed-parameter tractability and completeness III: some structural aspects of the W-hierarchy. In Ambos-Spies, K., Homer, S. and Schoning, U. (eds). *Complexity Theory.* Cambridge University Press, pp. 166–191.

Fellows, M. R. (1989) The Robertson-Seymour theorems: a survey of applications. *Contemporary Mathematics*, **89**, 1–18.

Fellows, M. R., Hallett, M. T. and Wareham, H. T. (1993) DNA physical mapping: three ways difficult. In Lengauer, T. (ed). *Proceedings: ESA'93 - European Symposium on Algorithms.* Lecture Notes in Computer Science no. 726. Springer-Verlag, Berlin, pp. 157–168.

Fellows, M. R. and Langston, M. A. (1992) On well-partial-ordering theory and its applications to combinatorial problems in VLSI design. *SIAM Journal of Discrete Mathematics*, **5**, 117–126.

Fraenkel, A. S. (1993) Complexity of protein folding. *Bull. Math. Biol.*, **55(6)**, 1199–1210.

Frauenfelder, H. and Wolynes, P. G. (1994) Biomolecules: where the physics of complexity and simplicity meet. *Physics Today*, **Feb.**, 58–64.

Gallant, J., Maier, D. and Storer, J. (1980) On finding minimal length superstrings. *JCSS*, **20(1)**, 50–58.

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

Goldberg, P. W., Golumbic, M. C., Kaplan, H. and Shamir, R. (1993) Four strikes against physical mapping of DNA. Technical report, Department of Computer Science, Tel Aviv University.

Golumbic, M. C. (1980), *Algorithmic graph theory and perfect graphs.* Academic Press, New York.

Golumbic, M. C., Kaplan, H. and Shamir, R. (1994) On the complexity of DNA physical mapping. Technical report, Department of Computer Science, Tel Aviv University. To appear, *Advances in Applied Mathematics.*

Gotoh, O. (1993) Optimal alignment between groups of sequences and its application to multiple sequence alignment. *Comput. Applic. Biosci.*, **9(3)**, 361–370.

Gusfield, D. (1993) Efficient methods for multiple sequence alignment with guaranteed error bounds. *Bull. Math. Biol*, **55(1)**, 141–154.

Hallett, M. T. and Wareham, H. T. (1994) A compendium of parameterized complexity results. *SIGACT News*, **25(3)**, 122–123.

Irving, R. W. and Fraser, C. B. (1992) Two algorithms for the longest common subsequence of three (or more) strings. In Apostolico, A., Crochemore, M., Galil, Z. and Manber, U. (eds). *Proceedings of the Third Annual Symposium on Combinatorial Pattern Matching (CPM).* Lecture Notes in Computer Science no. 644. Springer-Verlag, Berlin, pp. 214–229.

Jiang, T. and Li, M. (1993) On the complexity of learning strings and sequences. *Theor. Comput. Sci.*, **119**, 363–371.

Jiang, T. and Li, M. (1994a) Approximating shortest superstrings with constraints. To appear, *Theor. Comput. Sci.*

Jiang, T. and Li, M. (1994b) Optimization problems in molecular miology. In Du, D. Z. and Sun, J. (eds). *Advances in Optimization and Approximation.* Kluwer Academic Publishers, pp. 195–216,

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

Kaplan, H. and Shamir, R. (1993) Pathwidth, bandwidth, and completion problems to proper interval graphs with small cliques. Technical report, Department of Computer Science, Tel Aviv University.

Kaplan, H., Shamir, R. and Tarjan, R. E. (1994) Tractability of parameterized completion problems on chordal and interval graphs: minimum fill-in and physical mapping. To appear, *Proceedings of the 35th Annual IEEE Conference on the Foundations of Computer Science.*

Kececioglu, J. D. (1991) Exact and approximation algorithms for DNA sequence reconstruction. Technical report TR 91-26, Department of Computer Science, University of Arizona.

Kececioglu, J. D. (1993) The maximum weight trace problem in multiple sequence alignment. In *Proceedings of the Fourth Annual Symposium on Combinatorial Pattern Matching (CPM).* Lecture Notes in Computer Science no. 684. Springer-Verlag, Berlin, pp. 106–119.

Kruskal, J. B. and Sankoff, D. (1983) An anthology of algorithms and concepts for sequence comparison. In Sankoff, D. and Kruskal, J. B. (eds). *Time Warps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison.* Addison-Wesley, Reading, MA, pp. 265–310.

Maier, D. (1978) The complexity of some problems on subsequences and supersequences. *J. ACM*, **25(2)**, 322–336.

Martin, Y. C. (1991) Computer-assisted rational drug design. *Methods Enzymol.*, **203**, 597–613.

Michiels, F., Craig, A. G., Zehetner, G., Smith, G. P. and Lehrach, H. (1987) Molecular approaches to genome analysis: a strategy for the construction of ordered overlapping clone libraries. *Comput. Applic. Biosci.*, **3(3)**, 203–210.

Middendorf, M. (1994) More on the complexity of common superstring and supersequence problems. *Theor. Comput. Sci.*, **125(2)**, 205–228.

Ngo, T. J. and Marks, J. (1992) Computational complexity of a problem in molecular structure prediction. *Journal of Protein Engineering*, **5(4)**, 313–321.

Patthy, L. (1991) Exons – Original building blocks of proteins? *BioEssays*, **13(4)**, 187–192.

Pearson, W. R. and Miller, W. (1992) Dynamic programming algorithms for biological sequence comparison. *Methods Enzymol.*, **210**, 575–601.

Pevzner, P. A. (1989) l-Tuple DNA sequencing: computer analysis. *J. Biomol. Struct. & Dyn.*, **7(1)**, 63–73.

Pevzner, P. A. (1992) Multiple alignment, communication cost, and graph matching. *SIAM J. Appl. Math.*, **52(6)**, 1763–1779.

Pevzner, P. A. and Lipshutz, R. (1994) Towards DNA sequencing chips. In *19th Symposium on Mathematical Foundations of Computer Science.* Lecture Notes in Computer Science no. 841. Springer-Verlag, Berlin, pp. 143–158.

Pevzner,P. A., Lysov, Y. P., Khrapko, K. R., Belyavsky, A. V., Florentiev, V. L. and Mirzabejov, A. D. (1991) Improved chips for sequencing by hybridization. *J. Biomol. Struct. & Dyn.*, **9(2)**, 399–410.

Reeke Jr., G. N. (1988) Protein folding: computational approaches to an exponential-time problem. *Annual Reviews of Computer Science*, **3**, 59–84.

Southern E. M., Maskos, U. and Elder, J. K. (1992) Analyzing and comparing nucleic acid sequences by hybridization to arrays of oligonucleotides: evaluation using experimental models. *Genomics*, **13**, 1008–1017.

Sweedyk, E. and Warnow, T. (1994) The tree alignment problem is NP-hard. Technical report, Department of Computer Science, University of Pennsylvania.

Timkovskii, V. G. (1990) Complexity of common subsequence and supersequence problems and related problems. *Cybernetics*, **25**, 565–580. Translated from *Kibernetika*, **25**, 1–13.

Unger, R. and Moult, J. (1993) Finding the lowest free energy conformation of a protein is an NP-hard problem: proof and implications. *Bull. Math. Biol.*, **55(6)**, 1183–1198.

Wang, L. and Jiang, T. (1994) On the complexity of multiple sequence alignment. To appear, *Journal of Computational Biology.*

Wareham, H. T. (1993) On the computational complexity of inferring evolutionary trees. Technical report no. 9301, Department of Computer Science, Memorial University of Newfoundland.

Yue, K. and Dill, K. A. (1992) Inverse protein folding problem: designing polymer sequences. *Proc. Natl. Acad. Sci. USA*, **89**, 4163–4167.

Yue, K. and Dill, K. A. (1993) Sequence-structure relationships in proteins and copolymers. *Physical Review E*, **48(3)**, 2767–2778.