

Chapter 12

Problems

12.1 Introduction

Molecular biology has motivated many interesting combinatorial problems. A few years ago Pevzner and Waterman, 1995 [275] compiled a collection of 57 open problems in computational molecular biology. Just five years later a quarter of them have been solved. For this reason I don't explicitly say which of the problems below are open: they may be solved by the time you read this sentence.

12.2 Restriction Mapping

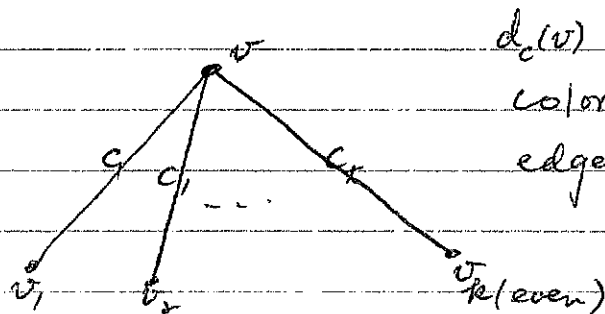
Suppose a DNA molecule is digested twice, by two restriction enzymes. The interval graph of resulting fragments is a *bipartite interval graph* (Waterman and Griggs, 1986 [361]).

Problem 12.1 Design an efficient algorithm to recognize a bipartite interval graph.

✓ **Problem 12.2** Let v be a vertex of even degree in a balanced colored graph. Prove that $d(v)$ edges incident to v can be partitioned into $d(v)/2$ pairs such that edges in the same pair have different colors.

Let $P = x_1 \dots x_m$ be a path in an l -colored balanced graph $G(V, E)$. A color c is *critical* for P if (i) it is different from the color of the last edge (x_{m-1}, x_m) in P and (ii) it is the most frequent color among the edges of $E \setminus EP$ incident to x_m (EP denotes the edge set of path P). The edges of the set $E \setminus EP$ incident to x_m and having a critical color are called the *critical edges*. A path is called critical if it is obtained by choosing a critical edge at every step.

Problem 12.3 Show how to use critical paths for constructing alternating Eulerian cycles.

Problem 12.2

$d_c(v) \leq \frac{1}{2} d(v)$ for each color c occurs in the edges incident to v

Let $c_1, c_2, \dots, c_k$ be the total number of colors occur around

v . Let $vv_i = e_i, i=1, 2, \dots, k$. Construct a new graph

$\tilde{G}$ with $V(\tilde{G}) = \{e_1, e_2, \dots, e_k\}$ and $\{e_i, e_j\} \in E(\tilde{G})$ if and only if

$c(e_i) \neq c(e_j)$. Now, since each color occurs at most $\frac{1}{2} d(v)$ times,

i.e., $\frac{k}{2}$. Hence, $\forall e_i \in V(\tilde{G}), \deg_{\tilde{G}}(e_i) \geq \frac{k}{2}$. $\tilde{G}$ has a

Hamilton cycle, and thus $\tilde{G}$ has a perfect matching, this

conclude the proof.

Can all vertices v be paired off?

The following problem asks for an analog of the BEST theorem for bicolored graphs.

Problem 12.4 Find the number of alternating Eulerian cycles in a bicolored Eulerian graph.

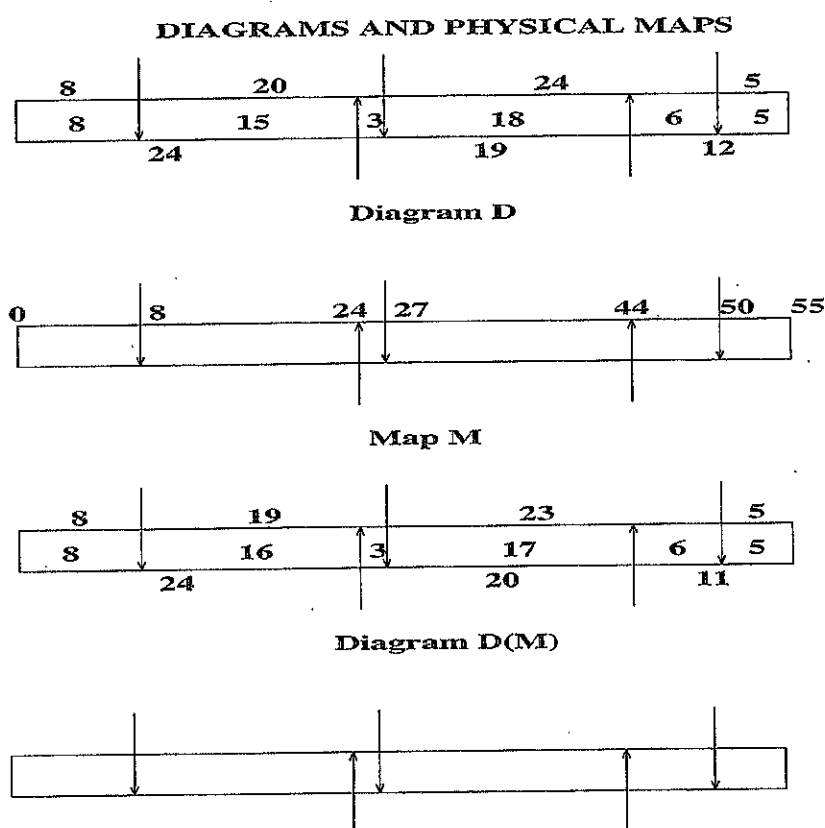


Figure 12.1: Diagrams and physical maps.

Most algorithms for the Double Digest Problem are based on generation and analysis of hypotheses about the order of restriction fragments in a physical map. Each such hypothesis corresponds to a *mapping diagram* D showing the order of sites and fragment lengths (Figure 12.1). Note that the coordinates of the sites are not shown in the mapping diagram. A physical map M provides information about both the order and the coordinates of the sites. Every physical map corresponds to a diagram $D(M)$ (Figure 12.1) with the lengths of the fragments corresponding to the distances between sites. The opposite is not true; not every diagram corresponds to a physical map. The question then arises of how to construct a physical map that best approximates the diagram. Let $D = (d_1, \dots, d_n)$ and $M = (m_1, \dots, m_n)$ be the lengths of all the fragments in diagrams D and

12.2. RESTRICTION MAPPING

$D(M)$ given in the same order. The distance between diagram D and map M is

$$d(D, M) = \max_{i=1, n} |d_i - m_i|.$$

For example, in Figure 12.1, $D = (8, 20, 24, 5, 24, 19, 12, 8, 15, 3, 18, 6, 5)$, $M = (8, 19, 23, 5, 24, 20, 11, 8, 16, 3, 17, 6, 5)$, and $d(D, M) = 1$. The *diagram adjustment problem* is to find a map within a shortest distance from a diagram:

Problem 12.5 Given a diagram D , find a map M minimizing $d(D, M)$.

Problem 12.6 Given two maps from the same equivalence class, find a shortest series of cassette transformations to transform one map into another.

A generalization of the Double Digest Problem is to have three enzymes A , B , and C and to get experimental data about the lengths of the fragments in *single* digestions A , B , and C , *double* digestions AB , BC , and CA , and *triple* digestion ABC . Such an experiment leads to the Multiple Digest Problem.

Problem 12.7 Find a physical map of three enzymes A , B , and C given provided six sets of experimental data (digestions A , B , C , AB , BC , CA , and ABC).

Problem 12.8 Characterize cassette transformations of multiple maps (three or more enzymes).

The Rosenblatt and Seymour, 1982 [289] PDP algorithm is pseudo-polynomial.

Problem 12.9 Does a polynomial algorithm exist for PDP?

Skiena et al., 1990 [314] proved that the maximum number $H(n)$ of strongly homometric sets on n elements is bounded by $\frac{1}{2}n^{0.6309} \leq H(n) \leq \frac{1}{2}n^{2.5}$. The upper bound seems to be rather pessimistic.

Problem 12.10 Derive tighter bounds for $H(n)$.

Problem 12.11 Prove that every 5-point set is reconstructible.

Problem 12.12 Design an efficient algorithm for the Probed Partial Digest Problem.

Problem 12.13 Derive upper and lower bounds for the maximum number of solutions for an n -site Probed Partial Digest Problem.

The input to the optical mapping problem is a 0-1 $n \times m$ matrix $S = (s_{ij})$ where each row corresponds to a DNA molecule (straight or reversed), each column corresponds to a position in that molecule, and $s_{ij} = 1$ if there is a cut in position j of molecule i . The goal is to reverse the orientation of a subset of molecules (subset of rows in S) and to declare a subset of the t columns “real cut sites” so that the number of ones in cut site columns is maximized (Karp and Shamir, 1998 [190]).

A naive approach to this problem is to find t columns with a large proportion of ones and declare them potential cut sites. However, in this approach every real site will have a reversed twin. Let $w(i, j)$ be the number of molecules with both cut sites i and j present (in either direct or reverse orientation). In a different approach, a graph on vertices $\{1, \dots, m\}$ is constructed and two vertices are connected by an edge (i, j) of weight $w(i, j)$.

Problem 12.14 *Establish a connection between optical mapping and the anti-symmetric longest path problem.*

12.3 Map Assembly

Problem 12.15 *Find the number of different interleavings of n clones.*

An interleaving of n clones can be specified by a sequence of integers $a_1 \dots a_n$, where a_i is the number of clones that end before the clone i starts. For example, the interleaving of nine clones in Figure 3.2 corresponds to the sequence 000112267. Not every sequence of integers $a_1 \dots a_n$ specifies a valid interleaving. Moreover, if a probe specifies a run $[i, j]$ of clones, this run implies the inequalities $a_j \leq i - 1$ (clones j and i overlap) and $a_{j+1} \geq i - 1$ (clones $i - 1$ and $j + 1$ do not overlap).

Problem 12.16 *Formulate the Shortest Covering String Problem with a given order of clones as a linear integer program, and solve it.*

Problem 12.17 *Let k be the maximal number of pairwise disjoint intervals in a collection of intervals $\mathcal{I}$ on a line. Prove that there exist k points on the line such that each interval in $\mathcal{I}$ contains at least one of these points.*

The intersection graphs corresponding to collections of arcs on a circle are called *circular-arc graphs*. If a collection of arcs on a circle does not cover some point x on the circle, then the *circular-arc graph* of this collection is an interval graph (cut the circle at x and straighten it out).

Problem 12.18 *Design an efficient algorithm to recognize circular-arc graphs.*

12.3. MAP ASSEMBLY

Problem 12.19 If N random clones of length L are chosen from a genome of length G , the expected fraction of the genome represented in these clones is approximately $1 - e^{-c}$, where $c = \frac{NL}{G}$ is the coverage.

In *cosmid contig mapping* (Zhang et al., 1994 [376]), clone overlap information is generated from hybridization data. A set of clones is placed on a filter for colony hybridization, and the filter is probed with a clone that has been radioactively labeled. This process produces overlap information as to which probes overlap with other clones. If only a subset of clones are used as probes, overlap information is not available between clones that are not probes. A graph is a *probe interval graph* if its vertex set can be partitioned into subsets P (clones used as probes) and N (clones not used as probes), with an interval assigned to each vertex, such that two vertices are adjacent if and only if their corresponding intervals overlap and at least one of the vertices is in P (McMorris et al., 1998 [235]).

Problem 12.20 Devise an algorithm to recognize probe interval graphs.

Inner Product Mapping (Perlin and Chakravarti, 1993 [263]) is a clone mapping approach that probes a set of radiation hybrid clones twice, once with BACs and once with STSs, to obtain a map of BACs relative to STSs. Inner Product Mapping requires two sets of data: a hybrid screening matrix with STSs and a hybrid screening matrix with BACs.

Problem 12.21 Given hybrid screening matrices with STSs and BACs, construct a map of BACs relative to STSs.

Elements $\pi_i \pi_j \pi_k$ for $1 \leq i < j < k \leq n$ form an *ordered triple* in a permutation $\pi = \pi_1 \dots \pi_n$. Let $\Phi(\pi)$ be a collection of all $\binom{n}{3}$ ordered triples for π . Radiation hybrid mapping motivates the following problem:

Problem 12.22 Given an arbitrary set T of ordered triples of an n -element set, find a permutation π such that $T \subset \Phi(\pi)$.

Elements $\pi_i \pi_j \pi_k$ form an *unordered triple* if either $1 \leq i < j < k \leq n$ or $1 \leq k < j < i \leq n$. Let $\Theta(\pi)$ be a collection of all unordered triples for π .

Problem 12.23 Given an arbitrary set T of unordered triples of an n -element set, find a permutation π such that $T \subset \Theta(\pi)$.

12.4 Sequencing

The simplest heuristic for the Shortest Superstring Problem is the GREEDY algorithm: repeatedly merge a pair of strings with maximum overlap until only one string remains. Tarhio and Ukkonen, 1988 [333] defined the *compression* of an SSP algorithm as the number of symbols saved by this algorithm compared to plainly concatenating all the strings.

Problem 12.24 *Prove that the GREEDY algorithm achieves at least $\frac{1}{2}$ the compression of an optimal superstring, i.e., $\frac{\text{GREEDY compression}}{\text{optimal compression}} \geq \frac{1}{2}$.*

A performance guarantee with respect to compression does not imply a performance guarantee with respect to length. Since an example for which the approximation ratio of GREEDY is worse than 2 is unknown, Blum et al., 1994 [37] raised the following conjecture:

Problem 12.25 *Prove that GREEDY achieves a performance guarantee of 2.*

Let $S = \{s_1, \dots, s_n\}$ be a collection of linear strings and $C = \{c_1, \dots, c_m\}$ be a collection of circular strings. We say that C is a *circulation* of S if every s_i is contained in one of the circular strings c_j for $1 \leq j \leq m$. The length of circulation $|C| = \sum_{j=1}^m |c_j|$ is the overall length of the strings from C .

Problem 12.26 *Find the shortest circulation for a collection of linear strings.*

Let $P = \{s_1, \dots, s_m\}$ be a set of *positive* strings and $N = \{t_1, \dots, t_k\}$ be a set of *negative* strings. We assume that no negative string t_i is a substring of any positive string s_j . A *consistent superstring* for (P, N) is a string s such that each s_i is a substring of s and no t_i is a substring of s (Jiang and Li, 1994 [180]).

Problem 12.27 *Design an approximation algorithm for the shortest consistent superstring problem.*

Short fragments read by sequencing contain errors that lead to complications in fragment assembly. Introducing errors leads to *shortest k -approximate superstring problem* (Jiang and Li, 1996 [181]):

Problem 12.28 *Given a set S of strings, find a shortest string w such that each string x in S matches some substring of w with at most k errors.*

Suppose we are given a set S of n *random* strings of a fixed size l in an A -letter alphabet. If n is large (of the order of A^l), the length of the shortest common superstring $E(S)$ for the set S is of the order n . If n is small, $E(S)$ is of the order $n \cdot l$.

12.5. DNA ARRAYS

Problem 12.29 Estimate $E(S)$ as a function of l , n , and A .

Given strings s and t , $\text{overlap}(s, t)$ is the length of a maximal prefix of s that matches a suffix of t .

Problem 12.30 Given a collection of i.i.d. strings $\{s_1, \dots, s_n\}$ of fixed length, find the distribution of $\max\{\text{overlap}(s_i, s_j) : 1 \leq i \neq j \leq n\}$.

Given a collection of reads $S = \{s_1, \dots, s_n\}$ from a DNA sequencing project and an integer l , the spectrum of S is a set S_l of all l -tuples from strings $s_1, \dots, s_n$. Let Δ be an upper bound on the number of errors in each DNA read. One approach to the fragment assembly problem is to first correct the errors in each read and then assemble the correct reads into contigs. This motivates the following problem:

Problem 12.31 Given S , Δ , and l , introduce up to Δ corrections in each string in S in such a way that $|S_l|$ is minimized.

12.5 DNA Arrays

Problem 12.32 Prove that the information-theoretic lower bound for the number of probes needed to unambiguously reconstruct an arbitrary string of length n is $\Omega(n)$.

Problem 12.33 Devise an algorithm for SBH sequence reconstruction by data with errors (false positive and false negative).

Given two strings with the same l -tuple composition, the distance between them is the length of the *shortest* series of transpositions transforming one into the other.

Problem 12.34 Devise an algorithm for computing or approximating the distance between two strings with the same l -tuple composition.

Problem 12.35 What is the largest distance between two n -letter strings with the same l -tuple composition?

Continuous stacking hybridization assumes an additional hybridization of short probes that continuously extends duplexes formed by the target DNA fragment and the probes from the sequencing array. In this approach, additional hybridization with a short m -tuple on the array $C(k)$ provides information about some $(k + m)$ -tuples contained in the sequence.

Problem 12.36 Given a spectrum S that does not provide an unambiguous SBH reconstruction, determine a minimum number of continuous stacking hybridization experiments needed to unambiguously reconstruct the target fragment.

Problem 12.37 Reconstruct the sequence of a DNA fragment given a spectrum S and the results of additional continuous stacking hybridizations.

A reduced binary array of order l is an array with memory $2 \cdot 2^l$ composed of all multiprobes of two kinds

$$\underbrace{\{W, S\}, \{W, S\}, \dots, \{W, S\}}_l \text{ and } \underbrace{\{R, Y\}, \{R, Y\}, \dots, \{R, Y\}}_l.$$

For example, for $l = 2$, the reduced binary array consists of 8 multiprobes: WW , WS , SW , SS , RR , RY , YR , and YY . Each multiprobe is a pool of four dinucleotides.

Problem 12.38 Compute the branching probability of reduced binary arrays and compare it with the branching probability of uniform arrays.

We call an array k -bounded if all probes in the array have a length of at most k .

Problem 12.39 Given m and k , find a k -bounded array with m (multi)probes maximizing the resolving power.

An easier version of the previous problem is the following:

Problem 12.40 Do binary arrays $C_{bin}(k-1)$ provide asymptotically the best resolving power among all k -bounded arrays?

Although binary arrays provide better resolving power than uniform arrays, an efficient algorithm for reconstruction of a DNA fragment from its spectrum over a binary array is still unknown.

Problem 12.41 Does there exist a polynomial algorithm for SBH sequence reconstruction by binary arrays?

The proof of theorem 5.6 considers the switch s_{i+1} at x, y and the switch s_i at z, u . The proof implicitly assumes that the sets of vertices $\{x, y\}$ and $\{z, u\}$ do not overlap.

Problem 12.42 Adjust the proof for the case in which these sets do overlap.

Let $\mathcal{L}$ be a set of strings. Consider a set E (*strings precedence data*) of all ordered pairs of different l -tuples such that they occur in some string from $\mathcal{L}$ in the given order but at arbitrary distances. Chetverin and Kramer, 1993 [66] suggested the *nested strand hybridization* approach to DNA arrays, which results in the following problem (Rubinov and Gelfand, 1995 [291]):

Problem 12.43 *Reconstruct $\mathcal{L}$ given strings precedence data.*

Two-dimensional Gray codes are optimal for minimizing the border length of uniform DNA arrays. However, for an *arbitrary* array, the problem of minimizing the overall border lengths of photolithographic masks remains unsolved.

Problem 12.44 *Find an arrangement of probes in an (arbitrary) array minimizing the overall border lengths of masks for photolithographic array design.*

12.6 Sequence Comparison

Fitting a sequence V into a sequence W is a problem of finding a substring W' of W that maximizes the score of alignment $s(V, W')$ among all substrings of W .

Problem 12.45 *Devise an efficient algorithm for the fitting problem.*

Problem 12.46 *Estimate the number of different alignments between two n -letter sequences.*

Problem 12.47 *Devise an algorithm to compute the number of distinct optimal alignments between a pair of strings.*

Problem 12.48 *For a pair of strings $v_1 \dots v_n$ and $w_1 \dots w_m$ show how to compute, for each (i, j) , the value of the best alignment that aligns the character v_i with character w_j .*

Problem 12.49 *For a parameter k , compute the global alignment between two strings, subject to the constraint that the alignment contains at most k gaps (blocks of consecutive indels).*

The k -difference alignment problem is to find the best global alignment of strings V and W containing at most k mismatches, insertions, or deletions.

Problem 12.50 *Devise an $O(kn)$ k -difference global alignment algorithm for comparing two n -letter strings.*

Chao et al., 1992 [64] described an algorithm for aligning two sequences within a diagonal band that requires only $O(nw)$ computation time and $O(n)$ space, where n is the length of the sequences and w is the width of the band.

Problem 12.51 *Can an alignment within a diagonal band be implemented with $O(w)$ space?*

Myers and Miller, 1988 [246] studied the following:

Problem 12.52 *Develop a linear-space version of global sequence alignment with affine gap penalties.*

Huang and Miller, 1991 [169] studied the following:

Problem 12.53 *Develop a linear-space version of the local alignment algorithm.*

In the space-efficient approach to sequence alignment, the original problem of size $n \times m$ is reduced to two subproblems of sizes $i \times \frac{m}{2}$ and $(n-i) \times \frac{m}{2}$. In a fast parallel implementation of sequence alignment, it is desirable to have a *balanced partitioning* that breaks the original problem into sub-problems of equal sizes.

Problem 12.54 *Design a space-efficient alignment algorithm with balanced partitioning.*

The score of a local alignment is not normalized over the length of the matching region. As a result, a local alignment with score 100 and length 100 will be chosen over a local alignment with score 99 and length 10, although the latter one is probably more important biologically. To reflect the length of the local alignment in scoring, the score $s(I, J)$ of local alignment involving substrings I and J may be adjusted by dividing $s(I, J)$ by the total length of the aligned regions: $\frac{s(I, J)}{|I| + |J|}$. The *normalized local alignment* problem is to find substrings I and J that maximize $\frac{s(I, J)}{|I| + |J|}$ among all substrings I and J with $|I| + |J| \geq k$, where k is a threshold for the minimal overall length of I and J .

Problem 12.55 *Devise an algorithm for solving the normalized local alignment problem.*

A string X is called a *supersequence* of a string V if V is a subsequence of X .

Problem 12.56 *Given strings V and W , devise an algorithm to find the shortest supersequence for both V and W .*

Let P be a pattern of length n , and let T be a text of length m . The *tandem repeat* problem is to find an interval in T that has the best global alignment with some tandem repeat of P . Let P^m be the concatenation of P with itself m times. The tandem repeat problem is equivalent to computing the local alignment between P^m and T , and the standard local alignment algorithm solves this problem in $O(nm^2)$ time.

Problem 12.57 Find an approach that solves the tandem repeat problem in $O(nm)$ time.

An alignment of circular strings is defined as an alignment of linear strings formed by cutting (linearizing) these circular strings at arbitrary positions.

Problem 12.58 Find an optimal alignment (local and global) of circular strings.

A local alignment between two different strings A and B finds a pair of substrings, one in A and the other in B , with maximum similarity. Suppose that we want to find a pair of (non-overlapping) substrings *within* string A with maximum similarity (*optimal inexact repeat problem*). Computing the local alignment between A and itself does not solve the problem, since the resulting alignment may correspond to overlapping substrings. This problem was studied by Miller (unpublished manuscript) and later by Kannan and Myers, 1996 [184] and Schmidt, 1998 [308].

Problem 12.59 Devise an algorithm for the optimal inexact repeat problem.

Schoniger and Waterman, 1992 [310] extended the range of edit operations in sequence alignment to include *non-overlapping* reversals in addition to insertions, deletions, and substitutions.

Problem 12.60 Devise an efficient algorithm for sequence alignment with non-overlapping reversals.

In the *chimeric alignment* problem (Komatsoulis and Waterman, 1997 [205]), a string V and a database of strings $\mathcal{W} = \{W_1, \dots, W_N\}$ are given, and the problem is to find $\max_{1 \leq i \neq j \leq N} s(V, W_i \oplus W_j)$ where $W_i \oplus W_j$ is the concatenation of W_i and W_j .

Problem 12.61 Devise an efficient algorithm for the chimeric alignment problem.

Problem 12.62 Show that in any permutation of n distinct integers, there is either an increasing subsequence of length at least $\sqrt{n}$ or a decreasing subsequence of length at least $\sqrt{n}$.

A *Catalan* sequence is a permutation $x_1 \dots x_{2n}$ of n ones and n zeros such that for any prefix $x_1 \dots x_i$, the number of ones is at least as great as the number of zeros. The n -th *Catalan* number C_n is the number of such sequences.

Problem 12.63 Prove the following:

- C_n is the number of standard Young tableaux with two rows of length n .
- C_n is the number of permutations $\pi \in S_n$ with a longest decreasing subsequence of length at most 2.
- C_n is the number of sequences of positive integers $1 \leq a_1 \leq a_2 \leq \dots \leq a_n$ such that $a_i \leq i$ for all i .

Problem 12.64 Prove the recurrence $C_{n+1} = C_n C_0 + C_{n-1} C_1 + \dots + C_0 C_n$.

Problem 12.65 Prove that the length of the longest decreasing subsequence of permutation π is the length of the first column of the Young tableau $P(\pi)$.

A subsequence σ of permutation π is k -increasing if, as a set, it can be written as

$$\sigma = \sigma_1 \cup \sigma_2 \cup \dots \cup \sigma_k$$

where any given σ_i is an increasing subsequence of π .

Problem 12.66 Devise an algorithm to find longest k -increasing subsequences.

Chang and Lampe [61] suggested an analog of the Sankoff-Mainville conjecture for the case of the edit distance $d(V, W)$ between n -letter i.i.d. strings V and W :

Problem 12.67 For random i.i.d. n -letter strings in a k -letter alphabet,

$$\frac{\text{Expectation}(d(V, W))}{n} = 1 - \frac{1}{\sqrt{k}} + o\left(\frac{1}{\sqrt{k}}\right).$$

Gusfield et al., 1994 [146] proved that the number of convex polygons in the parameter space decomposition for *global* alignment is bounded by $O(n^{2/3})$, where n is the length of the sequences. Fernandez-Baca et al., 1999 [102] studied the following:

Problem 12.68 Generate a pair of sequences of length n that have an order of $\Omega(n^{2/3})$ regions in the decomposition of the parameter space.

For a fixed-length alphabet, no examples of sequences with $\Omega(n^{2/3})$ regions in the parameter space decomposition are known.

Problem 12.69 Improve the bound $O(n^{2/3})$ on the number of regions in space decomposition for global alignment in the case of a bounded alphabet.

Problem 12.70 *Derive bounds for the expected number of regions in space decomposition for global alignment of two random sequences of length n .*

Problem 12.71 *Generalize bounds for the number of regions in space decomposition for the case of multiple alignment.*

Parameter space decomposition for local alignment usually contains more regions than parameter space decomposition for global alignment. Vingron and Waterman, 1994 [346] studied the links between the parametric sequence alignment and the phase transition. In this connection, it is interesting to study the parameter decomposition of logarithmic area.

Problem 12.72 *Derive bounds for the expected number of regions in space decomposition of logarithmic area for local alignment of two random sequences of length n .*

The Gusfield et al., 1994 [146] algorithm for parametric sequence alignment of two sequences runs in $O(R + E)$ time per region, where R is the number of regions in the parametric decomposition and E is the time needed to perform a single alignment. In the case of unweighted scoring schemes, $R = O(E)$, so the cost per region is $O(E)$. When one uses a weight matrix, little is known about R . Gusfield formulated the following:

Problem 12.73 *Estimate the number of regions in a convex decomposition in the case of weight matrices.*

Problem 12.74 *Devise a fast algorithm for space decomposition in the case of weight matrices.*

Since energy parameters for RNA folding are estimated with errors, it would be useful to study parametric RNA folding. For example, comparison of regions corresponding to cloverleaves for tRNA parameter space decomposition would provide an estimate of the accuracy of currently used RNA energy parameters.

Problem 12.75 *Develop an algorithm for parametric RNA folding.*

Let $S_n(\mu, \delta)$ be a random variable corresponding to the score ($\#$ matches $-\mu\#$ mismatches $-\sigma\#$ indels) of the global alignment between two random i.i.d. strings of length n . Arratia and Waterman, 1994 [14] defined $a(\mu, \delta) = \lim_{n \rightarrow \infty} \frac{S_n(\mu, \delta)}{n}$ and demonstrated that $\{a = 0\} = \{(\mu, \delta) : a(\mu, \delta) = 0\}$ is a continuous phase transition curve.

Problem 12.76 *Characterize the curve $a(\mu, \delta) = 0$.*

12.7 Multiple Alignment

Problem 12.77 *Devise a space-efficient algorithm for multiple alignment.*

Problem 12.78 *Devise an algorithm that assembles multiple alignments from 3-way alignments.*

Problem 12.79 *Construct an example for which the Vingron-Argos matrix multiplication algorithm requires $\Omega(L)$ iterations, where L is the length of sequences.*

Jiang and Li, 1994 [180] formulated the following:

Problem 12.80 *Can shortest common supersequences (SCSs) and longest common supersequences (LCSs) on binary alphabets be approximated with a ratio better than 2?*

One could argue that the notion of NP-completeness is somewhat misleading for some computational biology problems, because it is insensitive to the limited parameter ranges that are often important in practice. For example, in many applications, we would be happy with efficient algorithms for multiple alignment with $k \leq 10$. What we currently have is the $O((2n)^k)$ dynamic programming algorithm. The NP-completeness of the multiple alignment problem tells us almost nothing about what to expect if we fix our attention on the range of $k \leq 10$. It could even be the case that there is a linear-time algorithm for every fixed value of k ! For example, it would be entirely consistent with NP-completeness if the problem could be solved in time $O(2^k n)$.

The last decade has seen the development of algorithms that are particularly applicable to problems such as multiple alignment for fixed parameter ranges. We presently do not know whether the complexity obtained by dynamic programming is the “last word” on the complexity of the multiple alignment problem (Bodlaender et al., 1995 [38]). Mike Fellows formulated the following conjecture:

Problem 12.81 *The longest common subsequence problem for k sequences in a fixed-size alphabet can be solved in time $f(k)n^\alpha$ where α is independent of k .*

12.8 Finding Signals in DNA

Problem 12.82 *Describe a winning strategy for B in the best bet for simpletons.*

Problem 12.83 *Describe the best strategy for A in the best bet for simpletons (i.e., the strategy that minimizes losses).*

If a coin in the best bet for simpletons is biased (e.g., $p(0) > p(1)$), it makes sense for A to choose a word such as $0 \dots 0$ to improve the odds.

Problem 12.84 *Study the best bet for simpletons with a biased coin. Does B still have an advantage over A in this case?*

Problem 12.85 *Derive the variance of the number of occurrences of a given word in the case of linear strings.*

Problem 12.86 *Devise an approximation algorithm for the consensus word problem.*

The Decoding Problem can be formulated as a longest path problem in a directed acyclic graph. This motivates a question about a space-efficient version of the Viterbi algorithm.

Problem 12.87 *Does there exist a linear-space algorithm for the decoding problem?*

12.9 Gene Prediction

The spliced alignment algorithm finds exons in genomic DNA by using a related protein as a template. What if a template is not a protein but other (uninterpreted) genomic DNA? In particular, can we use (unannotated) mouse genomic DNA to predict human genes?

Problem 12.88 *Generalize the spliced alignment algorithm for alignment of one genomic sequence against another.*

Problem 12.89 *Generalize the similarity-based approach to gene prediction for the case in which multiple similar proteins are available.*

Sze and Pevzner, 1997 [332] formulated the following:

Problem 12.90 *Modify the spliced alignment algorithm for finding suboptimal spliced alignments.*

The “Twenty Questions Game with a Liar” assumes that every answer in the game is false with probability p . Obviously, if $p = \frac{1}{2}$, the game is lost since the liar does not communicate any information to us.

Problem 12.91 *Design an efficient strategy for the “Twenty Questions Game with a Liar” that finds k unknown integers from the interval $[1, n]$ if the probability of a false answer is $p \neq \frac{1}{2}$.*

Problem 12.92 Estimate the expected number of questions in the "Twenty Questions Game with a Liar" if the probability of a false answer is $p \neq \frac{1}{2}$.

Problem 12.93 Design experimental and computational protocols to find all alternatively spliced variants for a given genomic sequence.

The observation that PCR-based queries can be used to test a potentially exponential number of hypotheses about splicing variants leads to a reformulation of the above problem.

Problem 12.94 Given a graph $G(V, E)$ with a collection $\mathcal{C}$ of (unknown) paths, reconstruct $\mathcal{C}$ by asking the minimum number of queries of the form: "Does a collection $\mathcal{C}$ contain a path passing through vertices v and w from G ?"

Let S be a fixed set of probes, and let C be a DNA sequence. A fingerprint of C is a subset of probes from S that hybridize with C . Let G be a genomic sequence containing a gene represented by a cDNA clone C . Mironov and Pevzner, 1999 [241] studied the following fingerprint-based gene recognition problem:

Problem 12.95 Given a genomic sequence G and the fingerprint of the corresponding cDNA clone C , predict a gene in G (i.e., predict all exons in G).

12.10 Genome Rearrangements

Sorting by reversals corresponds to eliminating breakpoints. However, for some permutations (such as 563412), no reversal reduces the number of breakpoints. All three strips (maximal intervals without breakpoints) in 563412 are increasing.

Problem 12.96 Prove that if an unsigned permutation has a decreasing strip, then there is a reversal that reduces the number of breakpoints by at least one.

A 2-greedy algorithm for sorting π by reversals chooses reversals ρ and σ such that the number of breakpoints in $\pi \cdot \rho \cdot \sigma$ is minimal among all pairs of reversals.

Problem 12.97 Prove that 2-greedy is a performance guarantee algorithm for sorting by reversals.

In the case in which the sequence of genes contains duplications, sorting permutations by reversals is transformed into sorting words by reversals. For example, the shortest sequence of reversals to transform the word 43132143 into the word 42341314 involves two inversions: $43132143 \rightarrow 42313143 \rightarrow 42341314$.

Problem 12.98 Devise a performance guarantee algorithm for sorting words by reversals.

Kececioglu and Sankoff, 1994 [193] studied the bounds for the diameter $D(n)$ in the case of signed permutations and proved that $n - 1 \leq D(n) \leq n$. They also conjectured the following:

Problem 12.99 For signed circular permutations, $D(n) = n$ for sufficiently large n .

Problem 12.100 Characterize the set of "hard-to-sort" signed permutations on n elements such that $d(\pi) = D(n)$.

Problem 12.101 Improve the lower bound and derive an upper bound for the expected reversal distance.

Problem 12.102 Estimate the variance of reversal distance.

Gates and Papadimitriou, 1979 [120] conjectured that a particular permutation on n elements requires at least $\frac{19}{16}n$ reversals to be sorted. Heydari and Sudborough, 1997 [161] disproved their conjecture by describing $\frac{18}{16}n + 2$ reversals sorting the Gates-Papadimitriou permutation.

Problem 12.103 Find the prefix reversal diameter of the symmetric group.

Genomes evolve not only by inversions but by *transpositions* as well. For a permutation π , a *transposition* $\rho(i, j, k)$ (defined for all $1 \leq i < j \leq n + 1$ and all $1 \leq k \leq n + 1$ such that $k \notin [i, j]$) "inserts" an interval $[i, j - 1]$ of π between π_{k-1} and π_k . Thus $\rho(i, j, k)$ corresponds to the permutation

$$\begin{pmatrix} 1 & \dots & i-1 & \boxed{i \ i+1 \ \dots \ \dots \ j-2 \ j-1} & \boxed{j \ \dots \ k-1} & k & \dots & n \\ 1 & \dots & i-1 & \boxed{j \ \dots \ k-1} & \boxed{i \ i+1 \ \dots \ \dots \ j-2 \ j-1} & k & \dots & n \end{pmatrix}$$

Given permutations π and σ , the *transposition distance* is the length of the shortest series of transpositions $\rho_1, \rho_2, \dots, \rho_t$ transforming π into $\pi \cdot \rho_1 \cdot \rho_2 \cdot \dots \cdot \rho_t = \sigma$. *Sorting π by transpositions* is the problem of finding the transposition distance $d(\pi)$ between π and the identity permutation ι . Bafna and Pevzner, 1998 [20] devised a 1.5 performance guarantee algorithm for sorting by transpositions and demonstrated that the transposition diameter $D_t(n)$ of the symmetric group is bounded by $\frac{n}{2} \leq D_t(n) \leq \frac{3n}{4}$.

Problem 12.104 Find the transposition diameter of the symmetric group.

The well-known variant of sorting by transpositions is sorting by transpositions $\rho(i, i+1, i+2)$ where the operation is an exchange of adjacent elements. A simple bubble-sort algorithm solves this problem for linear permutations. Solving the problem for circular permutations is more difficult.

Problem 12.105 Design an algorithm for sorting circular permutations by exchanges of adjacent elements.

Problem 12.106 Every circular permutation can be sorted in $2\lceil \frac{n-1}{2} \rceil \lfloor \frac{n-1}{2} \rfloor$ exchanges of adjacent elements.

We represent a circular permutation as elements $\pi_1 \dots \pi_n$ equally spaced on a circle. Figure 12.2 presents circular permutations $\pi = \pi_1 \dots \pi_n$ and $\sigma = \sigma_1 \dots \sigma_n$ positioned on two concentric circles and n edges $e_1 \dots e_n$ such that e_i joins element i in π with element i in σ . We call such a representation of π and σ an *embedding*, and we are interested in embeddings minimizing the number C of *crossing edges*. Edges in an embedding can be directed either clockwise or counterclockwise; notice that the overall number of crossing edges depends on the choice of directions. For example, the embedding in Figure 12.2a corresponds to $C = 2$, while the embedding in Figure 12.2b corresponds to $C = 3$. An n -mer *direction vector* $\mathbf{v} = v_1, \dots, v_n$ with $v_i \in \{+1, -1\}$ defines an embedding by directing an edge e_i clockwise if $v_i = +1$ and counterclockwise otherwise.

For convenience we choose the “twelve o’clock” vertex on a circle to represent a “starting” point of a circular permutation. Choosing an element r as a starting point of π defines a *rotation* of π . For the sake of simplicity, we assume that $\sigma = 1 \dots n$ is the identity permutation and that the starting point of σ is 1.

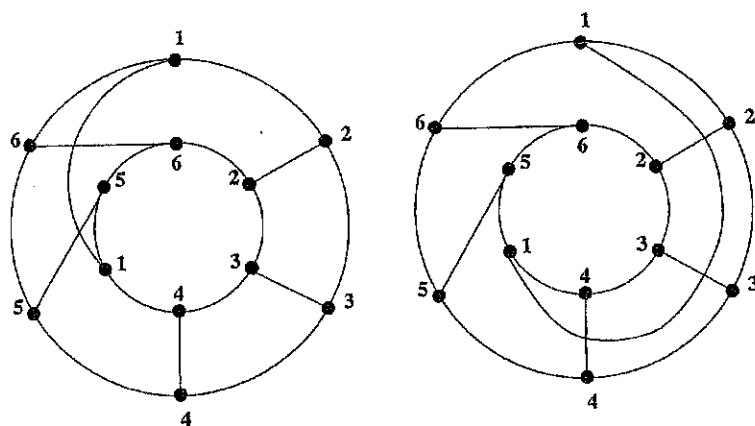


Figure 12.2: Crossing edges in embeddings.

Every rotation of π with r as a starting point and every vector $\mathbf{v}$ define an embedding with a number of *crossing edges* $C(r, \mathbf{v})$. Sankoff and Goldstein, 1989 [303] studied the following *optimal embedding* problem:

Problem 12.107 Find

$$\min_{r, \mathbf{v}} C(r, \mathbf{v}).$$

Let $d_{ij}(\pi)$ be the distance from element i to element j in permutation π counted clockwise. The length of a clockwise edge e_i is defined as $\hat{i} = (d_{1i}(\sigma) - d_{1i}(\pi)) \bmod(n)$, while the length of a counterclockwise edge is defined as $\tilde{i} = n - \hat{i}$. For a rotation r , define a canonical direction vector $\mathbf{v}(r) = (v_1(r) \dots v_n(r))$ according to the rule that $v_i(r) = +1$ if clockwise edge e_i is shorter than counterclockwise edge e_i (i.e., if $\hat{i} < \tilde{i}$) and -1 otherwise.

Problem 12.108 Prove

$$\min_r C(r, \mathbf{v}(r)) \leq \lceil \frac{n-1}{2} \rceil \lfloor \frac{n-1}{2} \rfloor.$$

Problem 12.109 Prove that for every $0 \leq r \leq n-1$, circular permutation π can be sorted in at most $C(r, \mathbf{v}(r))$ exchanges of adjacent elements.

An $\mathcal{A}$ -permutation on n elements is a permutation of $\{1, 2, \dots, n\}$ interspersed with letters from an alphabet $\mathcal{A}$. For example, 3aa1baa4a52b is an $\mathcal{A}$ -permutation of 31452 with $\mathcal{A} = \{a, b\}$. A reversal of an $\mathcal{A}$ -permutation is valid if its starting and ending elements coincide. An identity $\mathcal{A}$ -permutation is a permutation in which $\{1, 2, \dots, n\}$ appear in order (with arbitrary assignment of elements from $\mathcal{A}$).

Problem 12.110 Design a test deciding whether an $\mathcal{A}$ -permutation can be sorted by valid reversals.

Problem 12.111 Design an algorithm for sorting $\mathcal{A}$ -permutations by valid reversals.

12.11 Computational Proteomics

Mass-spectrometry has become a source of new protein sequences, some of them previously unknown at the DNA level. This raises the problem of bridging gap between proteomics and genomics, i.e., problem of sequencing DNA based on information derived from large-scale MS/MS peptide sequencing. This problem is complicated since peptides sequenced by mass-spectrometry may come in short pieces with potential ambiguities (such as transposition of adjacent amino acids and wild cards).

Problem 12.112 *Given a set of peptides (with ambiguities) from a given protein, design experimental and computational protocols to find the genomic sequence corresponding to this protein.*

Problem 12.113 *Design an algorithm for searching peptides with ambiguities in protein databases.*

Consider a mixture of (unknown) proteins subject to complete digestion by a protease (e.g., trypsin). This results in a collection of peptides ranging in length from 10 to 20 amino acids, and the problem is to decide which peptides belong to the same proteins and to reconstruct the order of peptides in each of the proteins. The mass-spectrometer is capable of (partial) sequencing of all peptides in the mixture, but it is typically unknown which peptides come from the same protein and what the order is of peptides in the proteins. Protein sequencing of protein mixtures is a problem of assembling peptides into individual proteins.

Problem 12.114 *Devise experimental and computational protocols for sequencing protein mixtures by mass-spectrometry.*

The spectrum graph approach to *de novo* peptide sequencing does not take into account internal ions and multiple-charged ions.

Problem 12.115 *Devise a peptide sequencing algorithm taking into account internal and multiple-charged ions.*

Let $M(P)$ be the set of masses of all partial peptides of peptide P . Using digestion of P by different non-specific proteases, one can obtain a set of experimentally measured masses of partial peptides $M \subset M(P)$.

Problem 12.116 *Given a set of masses $M \subset M(P)$, reconstruct P .*

Accurate determination of the peptide parent mass is extremely important in *de novo* peptide sequencing. An error in parent mass leads to systematic errors in the masses of vertices for C-terminal ions, thus making peptide reconstruction difficult. In practice, the offsets between the real peptide masses (given by the sum of amino acids of a peptide) and experimentally observed parent masses are frequently so large that errors in peptide reconstruction become almost unavoidable.

Problem 12.117 *Given an MS/MS spectra (without parent mass), devise an algorithm that estimates the parent mass.*