

Purdue University
Purdue e-Pubs

Department of Computer Science Technical
Reports

Department of Computer Science

1995

Sequence Alignment in Molecular Biology

Alberto Apostolico

Raffaele Fiancarlo

Report Number:
95-075

Apostolico, Alberto and Fiancarlo, Raffaele, "Sequence Alignment in Molecular Biology" (1995).
Department of Computer Science Technical Reports. Paper 1247.
<https://docs.lib.purdue.edu/cstech/1247>

This document has been made available through Purdue e-Pubs, a service of the Purdue University Libraries.
Please contact epubs@purdue.edu for additional information.

**SEQUENCE ALIGNMENT IN
MOLECULAR BIOLOGY**

**Alberto Apostolico
Raffaele Giancarlo**

**CSD TR-95-075
November 1995**

Sequence Alignment in Molecular Biology

Alberto Apostolico* Raffaele Giancarlo†
Purdue University and Università di Padova Università di Palermo

November 28, 1995

PURDUE CS TR 95-075

Abstract

Molecular biology is a computationally intense realm of contemporary science and faces some of the current grand scientific challenges. In its context, tools that identify, store, compare and analyze effectively large and growing numbers of bio-sequences are found of increasingly crucial importance. Biosequences are routinely compared or aligned, in a variety of ways, to infer common ancestry, to detect functional equivalence, or simply while searching for similar entries in a database. A considerable body of knowledge has accumulated on sequence alignment during the past few decades. Without pretending to be exhaustive, this paper attempts a survey of some criteria of wide use in sequence alignment and comparison problems, and of the corresponding solutions. The paper is based on presentations and literature given at the Workshop on Sequence Alignment held at Princeton, N.J., in November 1994, as part of the DIMACS Special Year on Mathematical Support for Molecular Biology.

*Dipartimento di Elettronica e Informatica, Università di Padova, Via Gradenigo 6/A, 35131 Padova, Italy, (39-49)828-7710; axa@art.dei.unipd.it; partially supported by NSF grant CCR-92-01078, by NATO grant CRG 900293, by the National Research Council of Italy, and by the ESPRIT III Basic Research Programme of the EC under contract No. 9072 (Project GEPPCOM).

†Dipartimento di Matematica, University of Palermo, Via Archirafi 34, 90123 Palermo, Italy; raffaele@altair.math.unipa.it; partially supported by MURST Grant "Algoritmi, Strutture di Calcolo e Sistemi Informativi"; part of this work was done while the author was visiting AT&T Bell Labs., Murray Hill, NJ., U.S.A..

1 Introduction

Classical taxonomy is based on the assumption that conspicuous morphological and functional similarities in species denote close common ancestry. Likewise, modern molecular taxonomy pursues phylogeny and classification of living species based on the conformation and structure of their respective genetic codes. It is assumed that DNA code presides over the reproduction, development and sustenance of living organisms, part of it (RNA) being employed directly in various biological functions, part serving as a template or blueprint for proteins. In these latter, following somewhat the dual principle of modern architecture, function follows form. The apparent functional, structural and sequence resemblance of proteins found in organisms of completely different morphology has further stimulated interest in a taxonomy based on sequence homology.

Two biomolecules are said to be homologous if their sequences are likely to be offsprings of a common ancestor sequence. However, ancestor sequences are not available, and homology is deduced from the similarity of existing sequences: the more two such sequences are similar, the more likely their homology is assumed to be.

Early studies on methods and tools for the automated analysis and comparisons of biosequences were stimulated by the identification of the aminoacid sequences of a number of proteins which occurred in the 1960s. Activity in this area has been growing ever since, and culminates with the considerable outburst of recent years, as more and more sequences accumulate from the genome sequencing of humans and other species. Despite some key conceptual acquisitions and substantial practical progress, we still do not have consolidated methods for every sequence alignment task. This should not come as a surprise, since the complexity of biosequence alignment is rooted in some of the subtlest and most delicate aspects of scientific inference.

In an attempt to capture genetically significant relationships among sequences, several criteria have been proposed as measures of sequence similarity. While they all need to relate in one way or another to the basic mechanisms of evolution, there is no simple way to test and validate any one of them. At the empirical level, such a validation is made impossible by the uniqueness of the evolutionary experiment. At the epistemological level, the notion of a class of similar objects does not rest on very firm grounds. For instance, a claim known as the "Theorem of the Ugly Duckling" [93] states that as long as all of the predicates characterizing the objects to be classified are given the same importance or "weight", then a swan will be found to be just as similar to a duck as to another swan. The reason for this apparent paradox is in the fact that classification as we experience it on an empirical basis is only possible to the extent that the various predicates characterizing objects are given nonuniform weights. Applied to the study of biomolecules, this means that we shall be able to build better and better alignment algorithms as we learn more and more about precisely those homologies that our algorithms seek to detect.

DNA molecules appear naturally to be repositories of "information" that is propagated through evolutionary history and used in replication and transcription mechanisms central to the life-cycle of cells and organisms. Quantitative definitions and measures of such information seem to be at the heart of sequence comparison methods, and have been attempted in various ways over the years. On one hand, this information seems to be relatable to Shannon's notion of uncertainty and may be measured as such (see, e.g., [29]). On the other, this information is used at the cellular level for the synthesis of structure (for function), hence for departure from chaos. While Shannon's classical measure applies to the information *in transmission*, structural information possibly *stored* in biosequences seems rather related to the notion of redundancy. Measuring structure in finite objects, however, presupposes the accomplishment of the "intuitive program" of "analyzing randomness as far as it is possible within the region of finite sequences" [74], as cultivated already by Von Mises [91] and other statisticians at the turn of the century. One of the deepest acquisitions in

this domain is linked to Kolmogorov's innovative approach to the definition of information [56] (see also [64]). In this approach --which seems reminiscent in a seductive way of the very mechanism of molecular evolution--, information (alternatively, conditional information) is measured by the length of the recorded sequence of zeroes and ones that constitute a program by which a universal machine produces one string from scratch (alt., from another string). The sobering conclusion, however, is that under this scheme there is no such thing as a finite random sequence: while a great many sequences of sufficiently large length tend to behave randomly in the limit, any short sequence exhibits some kind of regularity. It appears thus that we can measure and study randomness (whence also structure) in finite objects (see, in particular, [2]) only to the extent that we can legitimately privilege (i.e., assign a high weight to) certain regularities and neglect others, a principle that seems to be pendant, in syntactic pattern recognition, to the statistical paradox of the ugly duckling.

In conclusion, the difficulties inherent to sequence alignment lie in the process of forming the necessary educated biases, in the dynamical and interactive process of extracting the feature weights. Along the way, the practice of sequence alignment shall constantly oscillate between the risks of overlooking important structure and discovering any arbitrarily chosen kind of structure everywhere.

There is a number of biological motivations and contexts for computer alignment of molecular sequences, which results in a variety of methods and tools. There is global alignment of pairs of sequences that are globally related by common ancestry, local alignments of related sequence segments, multiple alignments of members of protein families. Some special techniques of self-alignments have also emerged recently as useful tools. Finally, and not entirely disjointly from the above, ancillary alignments are performed in data base searches, typically, in order to detect homology of proteins. The corresponding available computational methods have developed considerably since the pioneering work by M. O. Dayhoff [21] and others. Since 1982, *Nucleic Acids Research* has routinely devoted special issues of 600 pages or more to the latest developments in this area. By 1983, enough original material had accumulated on sequence comparison techniques to warrant dedication of a full volume [79]. Part of these developments occurred in parallel with algorithmics on strings, thus contributing some remarkable challenges to algorithmic design. In general, however, the two endeavors conserved rather distinctive individual flavors. On one hand, the computational biologist would be typically driven by the need to obtain some significant speed-up in the processing of a limited class of sequences on some specific machine or class of (commercial) machines. On the other, the algorist would be concerned with unveiling the combinatorial structure of a rigidly defined problem, in order to achieve a higher *asymptotic* efficiency in computations performed by an abstract machine. Historically, efficient algorithmics on strings and efficient computational methods for bio-sequence manipulations have found separate places in the literature, and in fact mostly addressed separate audiences. In general, molecular biologists are dismayed to find that formalization and computation involves almost invariably a number of abstractions and simplifications that denature the original problem at least in part. On the other hand, mathematicians and computer scientists incline towards an attitude that a problem that does not lend itself to a crisp formulation, or which reveals itself as trivial or intractable, is no longer a problem. While some of the negative trends of the past are successfully being reversed, the challenges that lay ahead of sequence alignment are still numerous and staggering.

Some of the focal contemporary issues are addressed below in this paper, the basic purpose of which is a better understanding of what the biologists expect from sequence alignment, how much of such an expectation has been fulfilled and how one could go about in defining future tasks. The paper implicitly reviews some past history, and tries to pin-point the major results and biological findings that led designers of sequence alignment to pursue some avenues of research

rather than others. Next, bio-mathematical theory of sequence alignment addresses, comparing various models and notions of similarity between sequences. Some emphasis is placed on the mechanisms presiding over the choices of parameters (e.g., weighting functions). This leads us to the study of the statistical tools that are used to establish the significance of alignments, and how well they model the underlying biological knowledge. As mentioned, when models are translated into computational tools, some of their subtleties are inevitably lost. We thus try to understand here what are the trade-offs between the sophistication of the theoretical model and the difficulties associated with their implementation. This is of particular interest in the case of multiple sequence alignment, perhaps the single most prominent instance of a computationally intensive alignment task where approximations, computational tradeoffs and significance are all subtly intertwined.

2 Sequence Alignment Issues and Early Results

Intuitively, Sequence Alignment consists of establishing how closely two or more sequences (strings over an alphabet) are related to each other. Closeness is usually quantified according to some similarity or distance measure. For example, the sequence *aba* is close to the sequence *abda*, since one can be obtained from the other by insertion or deletion of a symbol. From now on, we use the term string to refer to a sequence of symbols.

The aim of this section is to get acquainted with some of the issues in sequence alignment and to point out early work in this area. Most of those issues will be discussed at length in the remaining sections of the paper.

The earliest reference to a problem involving sequence alignment dates back to 1879 and it is a puzzle due to Lewis Carrol [17]. It works as follows: two English words, of equal length, are given and one is asked to transform one into the other by substituting one letter at a time. Insertion and deletion of letters are not allowed. For example, head and tall can be transformed into each other by going through the words heal, teal and tell. Intuitively, one can say that the two given words are at a distance of 4, since we need those many substitutions to go from one to the other. Later on, in the realm of coding theory, that simple notion of distance has been "invented again" by Hamming [34] and extensively used for the design of good codes for data transmission [28].

A more complete notion of distance between sequences was defined by Levenshtein in 1965 [60]. It can be described as follows. Given two strings, transform one into the other by using a sequence of the following operations: insertion, deletion, substitution of a symbol and exch. The most commonly used weights for DNA are the ones by [27]. For amino acids, many weight systems have been proposed (see for instance [26, 65, 75, 76, 84]). However, for a long time the ones by [22] have been the standard and, very recently, the ones by [37] seem to be superior for the alignment of distantly related proteins [38].

A very important aspect of sequence alignment is to establish how meaningful a given alignment is. This basic question applies both to global and to local alignments. Informally, it can be stated as follows: is a given alignment of two sequences due to chance? Answers to this question involve the investigation of many statistical aspects of sequence alignment such as the expected length of an alignment or the expected distribution of scores of alignments. The mathematical tractability of those statistical questions seems to depend on two things: what kind of alignment we are considering and the weights that we assign to the edit operations. Early results in this area are reported in [19, 49, 86]. A closely related issue is a quantification of the sensitivity and selectivity of sequence alignment methods. Sensitivity is the ability to detect distant evolutionary relationships between two strings, while selectivity is the ability to avoid the assignment of high scores to unrelated strings. In order to evaluate an alignment method with respect to those two parameters, one needs

criteria to establish whether a meaningful alignment has been found or missed.

Obviously, also the time required to compute an alignment of two strings is very important. As for the statistical significance of an alignment, the efficiency seems to depend on the chosen alignment method and the properties satisfied by the weights. It is interesting to point out that over the years the following trend has taken place. As strings to be aligned got longer, the design of algorithms computing the alignment "shifted" from the basic Needleman and Wunch method [68] (which computes an optimal alignment) to methods that give up the optimality of the computed alignment for speed. Two excellent examples are the algorithms reported in [5, 62, 97, 98]. For the interested reader, an overview of some algorithmic issues related to this trade-off speed-optimality as well as to the impact of weights on the computational speed can be found in [30].

Alignment methods for two strings can be used for data base searches. That is, we are given a data base of proteins and we want to establish whether a query string is biologically related to strings in the data base. Usually, that involves the computation of a local similarity between the query string and each string in the data base. The strings having a similarity score exceeding some threshold are usually reported and further screened to assess the biological meaningfulness of the method. Statistical knowledge about score distributions or empirical knowledge is used for that screening.

As the knowledge about biological sequences grows, one tries to ask more complex questions about the "closeness" of two or more sequences. One such type of question deals with the problem of finding approximate repeated patterns within each string. Those approximate repeated patterns seem to be associated with several important properties of DNA and proteins. For instance, parts of chromosomes are made of repeated patterns. A fixed number of those repeated patterns is lost when the cell divides. So, it seems that those repeated patterns are some kind of clock "ticking" the number of divisions that the cell can still undergo. The first algorithmic studies in this area are due to Miller [66] and Landau and Schmidt [57].

We will see an example of this when we discuss the problem of locating tandem repeats in a string [81].

Another important and very active area of Sequence Alignment is the one that deals with the alignment of multiple sequences. There are many ways in which this problem can be stated and, in what follows, we will discuss only a few of them. For instance, Sankoff stated the following problem, now referred to as multiple alignment under an evolutionary tree [77]. We are given a set of strings (say, proteins) labeling the leaves of a given tree. We want to find ancestral strings to label the internal nodes of the tree and such that a given score function is minimized. Intuitively, one is trying to establish whether a set of strings have common ancestors and how those ancestors look like. Sankoff also provided algorithms for this problem [77]. Another version of multiple sequence alignment is the one that asks for the longest common subsequence of a set of strings [39].

For multiple sequence alignment problems one faces the same difficulties and issues already mentioned for the alignment of two sequences, i.e., choice of weights, statistical significance, etc., except that here things are much harder. For instance, many multiple sequence alignment problems are NP-Hard (see for instance [92]), or too time consuming (see [41]). Also in the case of multiple sequence alignment, there is a trend to develop fast algorithms that trade-off optimality of the required solution for speed (see for instance [11]). However, in some cases, even the computation of approximate solutions is hard (see for instance [92]).

A string of DNA encodes the genome (or parts of) a living organism. The notion of distance between strings that uses the edit operations previously defined models evolution through a set of very local changes when applied to genomes. Although such a distance is appropriate to describe evolution of small parts of a genome (for instance, a substring of a string describing a chromosome), it shows limitations in quantifying the evolutionary distance between two entire genomes or two

entire chromosomes within two genomes. Indeed, evolution at the level of the genome seems to involve non-local, large scale operations, which can rearrange a whole segment of a chromosome in one evolutionary event. Those non-local changes are hard to detect using a distance based on very local edit operations (see for instance discussions in [32, 50, 69, 78, 80, 96]). In order to overcome this difficulty, one tries to “compare” two genomes by using more structured information. The idea is the following. Assume that we can write both genomes we want to compare as two strings of genes. Now, molecular biology suggests that if the two genomes are related, one should be obtainable from the other by a suitable rearrangement of its genes. This approach (gene rearrangement) has been pioneered by Palmer and Herbon [69] even though the earliest application of gene rearrangement to genome comparison seems to date back to 1938 [23]. From the combinatorial point of view, gene rearrangement requires the definition of new distances between genomes (strings of DNA). Moreover, it gives rise to a set of very interesting and challenging combinatorial problems. In this context, the first formulation of a new distance and of an algorithm computing it seems to be due to [96]. Very recently, there has been quite a bit of work, initiated by Kececioğlu and Sankoff [54], for the formal investigation of the computational complexity of various distances between genomes. Later on, we will briefly review the state of the art.

3 Two Basic Sequence Alignment Methods

In this section we introduce two methods that are used to compute similarity between two proteins, i.e., two strings $x[1 : n]$ and $y[1 : m]$ over the alphabet of amino acids. Those methods compute two (differently defined) local similarity scores. The first one is a specialized version of the second.

MSP. It computes an extremely simple notion of alignment in which no insertions and deletions of symbols are allowed. Consider two substrings $x[i : i + k]$ and $y[j : j + k]$ of x and y , respectively. We can align those two substrings by pairing $x[i]$ with $y[j]$, $x[i + 1]$ with $y[j + 1]$ and so on. We can define a score for that alignment. Indeed, let $s(a, b)$ be the weight of “pairing” a with b , where a and b are letters over the alphabet Σ of amino acids. s can be thought of as a matrix, which we refer to as the substitution matrix. The score of aligning two substrings (of equal length) of x and y is given by the sum of the weights of the letters paired together. An alignment is locally optimal if and only if its score cannot be improved by either extending both substrings or by shortening both of them. The Maximal Segment Pair is the best locally optimal alignment and it is taken as the (local) alignment of the two strings. In some cases, one may be interested in all locally optimal alignments.

Other methods of this kind have also been proposed (see for instance [9, 10, 51, 83]). Moreover, this simple method is the base for the BLAST program, one of the fastest programs for local sequence alignment [5].

SW-Local ([45, 82, 85]). It computes local alignments of two substrings of x and y , but this time insertion and deletion of symbols are allowed. For each pair of prefixes $x[1 : i]$ and $y[1 : j]$ we are interested in finding the best suffixes of those two prefixes that can be aligned with each other using the full set of edit operations. That can be done by means of the following dynamic programming recurrence (obtained by Smith and Waterman [85]):

$$M_{i,j} = \max(M_{i-1,j-1} + s(x[i], y[j]), M_{i-1,j} + \delta, M_{i,j-1} + \delta, 0) \quad (1)$$

where δ is the weight of deleting or inserting a symbol, $1 \leq i \leq n$ and $1 \leq j \leq m$. It is chosen to be a negative number. The initial conditions of the recurrence are given by $M_{i,0} = 0$ and $M_{0,j} = 0$.

The alignments computed by the above recurrence are locally optimal. Among those, we choose the best, i.e., the one corresponding to $\max(M_{i,j})$, $1 \leq i \leq n$ and $1 \leq j \leq m$. We remark that a slight change in (1) and in its initial conditions yields a method for the computation of the global similarity between two strings. SW-local (with modifications by Gotoh [31]) is the base of the SSEARCH program for local alignment.

4 The Structure and Choice of Weights

In the two alignment methods we have outlined in the previous Section, δ and s are the weights corresponding to insertion/deletion and substitution of a symbol, respectively. The numeric values that one chooses for those weights are extremely important in order to obtain alignments that are biologically meaningful. There are two natural questions that one can ask about weights. What is the structure of weights, i.e., is there any mathematical methodology that can help us to obtain numeric values for weights that turn out to be useful for molecular biology? How do we actually compute those weights? The current state of the art provides partial answers to those questions and many of the results depend on whether we want to use weights for local or global alignment and on which method we are using. In what follows, we will first present results for the MSP method. So, we will discuss mainly substitution matrices in the local alignment context and, unless otherwise stated, an alignment of two strings is the one obtained by MSP. Then, we will consider the problem of how to choose gap and substitution weights for SW-local, since it is a good representative of the class of methods allowing gaps.

4.1 The Structure of Substitution Matrices

When one looks for optimal local alignments of two strings, one is trying to discover whether parts of those two strings are related to each other through evolution. The scores (and therefore the weights of substitutions) give a quantitative measure of that relationship. So, the weights should be designed to discriminate biologically meaningful alignments from ones due to chance. In very loose and intuitive terms, two strings can always be obtained one from the other through some amount of “evolutionary change”. So, when we align those two strings, it seems reasonable to use the weights that best capture that amount of “evolutionary change”. Unfortunately, that quantity is not known to us. However, in order to investigate the “structure of weights”, let us assume that it is known. Moreover, to simplify notation, we assume that all letters of the alphabet are integers in $[1, |\Sigma|]$.

Let p_i be the probability that symbol i appears in a randomly chosen string from Σ^* . We refer to the p_i 's as the background probability distribution of the symbols of the alphabet. Let $q_{i,j}$ be the probability that, for the amount of evolutionary change we have fixed, symbol i is substituted by symbol j . We refer to $q_{i,j}$'s as the target probability distribution of two symbols of the alphabet substituting each other. It has been shown [47] that the best weights one can use to capture the given amount of evolutionary change are of the form

$$s(i, j) = \frac{\log(q_{i,j}/p_i p_j)}{\lambda} \quad (2)$$

where λ is a normalization constant. A few remarks are in order. As already stated in Section 2, many substitution matrices have been proposed, even before (2) was discovered. Most of them are log-likelihood matrices and therefore have the form prescribed by (2). This is intuitively appealing, since $s(i, j)$ gives a “measure” of how much the probability of the event “symbol i substitutes

symbol j " differs from chance. Equation (2) is important because it points out that, at least for MSP alignments, that intuition is indeed correct.

There are other requirements that those substitution weights must satisfy. We will state and justify them. There must be at least two letters in Σ for which their weight is positive. Since we would like to assign positive scores to locally optimal alignments, that constraint avoids to consider empty alignments (of zero score). Consider the weight of two randomly chosen symbols, then its expected value must be negative. If the expected weight of two randomly chosen letters would be positive, extending two substrings in an alignment as far as possible would tend to increase the score of that alignment (so the definition of locally optimal alignment would be meaningless).

4.2 How to Compute Substitution Matrices

We now turn to the problem of how to actually determine the numeric values for substitution matrices. The hard part is the computation of the target probabilities in (2). We will briefly discuss two methods, one due to [22] and the other one due to [37].

The PAM Matrices. Dayhoff et al. [22] use a stochastic process to model evolution of proteins: at each discrete time instant, a symbol (an amino acid) has a certain fixed probability of substituting another symbol. They also define a unitary measure of evolutionary change as a Point Accepted Mutation (PAM for short): one PAM is defined as the amount of evolutionary change required for 1% of all symbols to be substituted by other symbols. In other words, in a given string, one PAM expresses 99% conservation and one symbol substitution per 100 symbols. Let M_1 be the stochastic matrix giving the target probability distribution at one PAM. That is, $M_1[i, j] = q_{i,j}$ at one PAM. M_1 is a matrix that has been experimentally determined by studying a group of closely related proteins [22]. $M_n = (M_1)^n$ is the target probability distribution at n PAM's.

Now, for a given evolutionary distance (expressed in PAM's), we can use the corresponding M matrix to compute the substitution matrix as given by (2). Actually, one has a family of matrices from which to obtain substitution matrices. Those target distribution matrices are simply referred to by the PAM distance corresponding to them. So, PAM 250 is the matrix corresponding to an evolutionary distance of 250 PAMs.

One drawback of the method just described to compute weights is that, for distantly related strings, one infers target probabilities from the target probabilities of closely related strings. That may lead to inaccuracies since we are not estimating the target probabilities directly from samples of distantly related strings. Henikoff and Henikoff [37] have recently proposed a different method to obtain target probabilities for distantly related strings. The idea is to infer those probabilities from a large set of representative families of proteins. Here we outline the main ideas of the method.

The BLOSUM Matrices. Since $|\Sigma| = 20$ for the alphabet of amino acids, there are 210 distinct pairs of symbols (i, j) , each of which corresponds to a substitution of symbol i with j . (Here (j, i) is considered to be same as (i, j)). The target probabilities are a normalized estimate of the frequencies with which one can find a substitution (i, j) . We estimate those frequencies and therefore the corresponding probabilities using a data base of proteins. That is done as follows. The proteins in the database are divided into blocks, according to the criteria outlined in [37]. A block is a set of s strings, each of length w . So, we can think of a block as an $s \times w$ matrix. For each column of that matrix, we count how many times symbols i and j appear in that column and we increase the frequency estimate of that pair of symbols accordingly. (Each such an occurrence corresponds to a substitution of a symbol of a string into another symbol in a different string in the block.) The process is repeated for all blocks.

An important feature of this method is the fact that one can cluster closely related proteins that are in the same block. Such a clustering avoids that closely related proteins give a large contribution to the target frequencies (we want to obtain target frequencies that “capture” similarity between distantly related proteins). The idea is the following. Let us assume that we want to reduce the contributions to the target frequencies of proteins that are at least 80% similar. Then, for each block, we cluster together all proteins that are at least 80% similar and each of those clusters is now one string in the block. We use those new blocks to estimate target frequencies. We remark that the similarity of proteins within blocks is obtained by other methods [37].

The above approach gives rise to a family of target probability distribution matrices, the BLOSUM family. Each BLOSUM matrix has also a number attached to it that refers to the percentage that one has used for clustering. For instance, BLOSUM 62 is the matrix obtained by clustering together, in each block, proteins that are at least 62% similar. From each BLOSUM matrix, we can obtain substitution matrices again using (2).

4.3 How to Choose Substitution Matrices

When we want to align two strings using MSP, we need to choose a substitution matrix. From what we have said above, we should use the substitution matrix that best captures the evolutionary change transforming one string into the other. Unfortunately, that distance is not known to us, even though we may have some idea about it. Up until recently, one would resort to use the PAM 250 matrix since experimentally it seemed the best suited to align distantly related strings. Recently, Altschul [3] has come up with a very interesting interpretation of substitution matrices in information theoretic terms. That interpretation is useful both in choosing a substitution matrix from a family of matrices (like the PAM or BLOSUM families) and in comparing the performance of matrices from different families, i.e., the ability of substitution matrices to yield biologically meaningful alignments. As for sequence alignment methods, performance of substitution matrices can be assessed by estimating their sensitivity and selectivity.

Let $H = \sum_{i,j} q_{i,j} \log_2(q_{i,j}/p_i p_j)$ be the relative entropy of the target and background distributions. We associate that quantity to a substitution matrix with target probabilities $q_{i,j}$'s.

Now, fix two strings for which we want to compute an MSP alignment and assume that $q'_{i,j}$ is the target probability distribution for the alignment of those two strings. Let H' be the same as H but with the new target probability distribution.

Notice that H' is the average score per symbol substitution in that alignment. Since H' is also a measure of information, we can interpret it as the average information (in bits) per position that is needed to distinguish that alignment from chance. In other words, it gives, for each position of the alignment, the amount of information that we need to discriminate between the target and the background probability distributions. Now, to align the two chosen strings, we should choose a substitution matrix with relative entropy close to H' . Since we do not know H' , we have to resort to approximate it. That is done as follows.

Altschul [3] has shown that, in order for the score of an MSP alignment to be meaningful, its score must be of at least $\log N$ bits, where N is the product of the lengths of the two strings being aligned. Now assume that, for the two strings we want to align, we expect the length of the best local alignment to be f (this value is usually determined by resorting to heuristic knowledge). So, the amount of bits each position must contribute to the score of that alignment is $\log N/f$. Therefore, we choose a substitution matrix with relative entropy close to $\log N/f$. Based on this criterion, Altschul [3] provides a set of guidelines on which PAM matrices to use in which contexts. Further studies are reported in [4, 87].

The relative entropy of a substitution matrix is also very important to establish a criterion

on which to base the comparison of the performance (sensitivity and selectivity) of two different substitution matrices. For instance, the PAM family of matrices has been obtained in a totally different way from the BLOSUM family and there seems to be no obvious relationship between members of the two families. The question is how to compare those two families of matrices (the performance of the PAM 250 matrix should be compared with BLOSUM 62 or with another matrix?). Using relative entropy, it seems reasonable to compare the performance of two matrices that have the same entropy. Using this criterion, Henikoff and Henikoff [37] have compared those two families of matrices and they showed that the BLOSUM matrices seem to perform much better than the PAM matrices. Additional extensive studies are reported in [38]. For completeness, we point out that those experimental studies have been carried out for local alignment methods that allow gaps.

4.4 Weights for Alignments with Gaps

Here we consider the problem of choosing weights for global or local alignment methods when both gaps and substitutions are allowed.

The weights consist of a substitution matrix and a gap penalty δ . The choice of the gap penalties is by trial and error. A typical choice is -12 for the first symbol in the gap (the first insertion or deletion in a series) and -4 for the remaining gaps in the series (see for instance [71]). It is an open problem to determine a provably optimal set of values for gap penalties.

The substitution matrix is usually chosen to be a log-likelihood matrix, i.e., a matrix whose entries have the form prescribed by (2). So, any of the matrices described in Section 4.2 can be used. Unfortunately, that choice is based on heuristics only. That is, so far, there is no theory stating that a substitution matrix for SW-local should be of that form. Moreover, it has been argued in [3] that a result analogous to equation (2) may not be possible to prove for the case of alignments with gaps.

For the time being, one resorts to extensive experimentations to establish how well a set of weights performs. In this context, it is important to come up with good methodologies on which to base the design of those experiments. Some steps in that direction have been taken (see for instance [37, 70]). We will see one of those methodologies in Section 5.

5 Sensitivity and Selectivity of Alignment Methods

Good algorithms for sequence alignment must show a good balance between sensitivity and selectivity. As already mentioned, sensitivity is the ability to detect distant evolutionary relationships between two strings, while selectivity is the ability to avoid the assignment of high scores to unrelated sequences. In order to evaluate an alignment method with respect to those two parameters, one needs criteria to establish whether a meaningful alignment has been found or missed.

One criterion is to establish the statistical significance for the score of a given alignment. For MSP, one can phrase the problem as follows. Consider two random protein strings, i.e., they satisfy the probabilistic assumptions stated in Section 4.1. How many distinct locally optimal alignments with score at least S are expected to occur simply by chance? Karlin and Altschul [47] have shown that such a number is well approximated by the formula

$$KN \exp^{-\lambda S} \quad (3)$$

where λ is the same normalization constant of equation (2), N is the product of the lengths of the strings we are aligning and K is a parameter that can be explicitly computed [47, 48]. Once that

we know the score S of a locally optimal alignment, we can use (3) to establish how meaningful it is (the smaller the number given by (3), the more meaningful the alignment is).

For the more general case of global or local alignment between two strings, where both gaps and substitutions are allowed, e.g., alignments computed by SW-local, it would be interesting to obtain some statistical theory about the expected distribution of optimal scores. For the time being, the only available results are for some important special cases (see [58, 95]). Because of that lack of knowledge, Monte Carlo techniques are sometimes used to establish how likely it is for a given alignment score to be due to chance [24, 62, 72].

Another approach that has been recently proposed is to come up with a methodology for the empirical assessment of the selectivity and sensitivity of alignment methods. Pearson [71] has proposed one such a methodology and he has applied it to the comparison of some alignment programs for data base searches. We will briefly outline the main points of that methodology.

The sensitivity and selectivity of a method is quantified by two parameters. One is the number of strings in the data base that are biologically related to the query string and that were "missed" by the method during a search (that measures sensitivity). The other is the number of strings in the data base that are biologically unrelated to the query string and that were "found" by the method during a search (that measures selectivity). Moreover, sensitivity and selectivity of an alignment method is evaluated with respect to the SSEARCH program.

As for establishing when a string in the data base has been found or missed, Pearson has suggested several statistical criteria [71]. In principle, those criteria would need the a priori knowledge of the statistical distribution of scores. That knowledge is experimentally precomputed as follows. Using previously available knowledge, the data base is conceptually divided into two sets of strings: the ones related to a sample query string and the ones that are not related. Using SSEARCH and a set of weights for the edit operations, the scores between the sample string and each unrelated string is computed. That gives a distribution of scores for unrelated sequences. The same is done for the set of related sequences.

Now a method is evaluated by running many experiments using different criteria to establish when it has found or missed a string in the data base. The method uses the same weights as the ones used by SSEARCH to precompute statistical information.

We remark that the above experimental methodology (or similar ones-see [37]) can also be used to evaluate the performance of substitution matrices. Indeed, one fixes the method once and for all and the experiments are performed by changing the matrices.

6 Self-Alignments

Sometimes it is important to locate repeated patterns in a string $x[1 : n]$. The two patterns that seem to be of most biological interest are non-overlapping repeats and tandem repeats.

Consider two substrings $x[j : r]$ and $x[\ell : t]$, with $\ell \geq r$. Let $Score(x_r^j, x_t^\ell)$ be the similarity score between those two strings. That is, the sum of the weights of the operations transforming one string into the other. Such a sequence of operations gives also an alignment of $x[j : r]$ with $x[\ell : t]$. We refer to that alignment as a repeat. The repeat is a tandem repeat if $\ell = r$. To keep things simple, let us assume that, among all possible repeats, we are interested in finding the best one. That is, the one of maximum score. This problem has been proposed by Miller [66], who also gave a practical algorithm solving it. The heart of the algorithm is the dynamic programming recurrence (1) that is used for local alignment. However, the computation of that recurrence needs to be modified to account for the facts that we are aligning a string with itself and that we are interested in non-overlapping regions of similarity. Kannan and Myers [46] found an interesting

way to combine some of the ideas in [66] with ideas by Apostolico et al [7] (see also [1]) for the parallel computation of the alignment between two strings. The algorithm they devised has an $O(n^2 \log^2 n)$ time performance.

Recently, Schmidt [81] has made important contributions in this area in several directions. She generalized the notion of locally optimal alignment for strings (the one due to Sellers [82]) to locally optimal (non-overlapping) repeats and to locally optimal tandem repeats. Then, she devised an $O(n^2 \log n)$ time algorithm that computes all those locally optimal repeats. The data structures she devised may be useful in other sequence alignment contexts. Intuitively, those data structures support fast answers to queries on shortest paths in grid graphs associated with the dynamic programming recurrence (1). For completeness, we mention that a simplified version of the problem of finding all locally optimal tandem repeats has been considered in [57].

7 Multiple Sequence Alignment

Given a set of strings x_1, x_2, \dots, x_k over an alphabet Σ , a multiple alignment for those strings is a two-dimensional matrix of k rows satisfying the following conditions. The entries of A are either symbols from the input alphabet or the empty symbol λ (the identity under concatenation), and concatenating the entries in row i of A yields the string x_i . Given a cost function d , defined on the columns of A , an optimal alignment is one that minimizes the sum of the costs on all columns of a multiple alignment matrix A . Formally, we want a multiple alignment matrix $A = (a_{ij})_{1 \leq i \leq k}$ that minimizes

$$\sum_j d(a_{1j}a_{2j}\dots a_{kj}). \quad (4)$$

A little reflection reveals that there is no unique way to make the problem statement precise, as the cost function d could be specified in several ways that are meaningful from a biological perspective, or necessary from a computational perspective, or both. Correspondingly, multiple alignment comes in several flavors. If d consists of tallying the number of distinct symbols in each column, then the output consists of a shortest common supersequence (SCS) of the input strings. If d is chosen so that $d(a_{1j}a_{2j}\dots a_{kj}) = -1$ when all symbols in a column are identical and is 0 otherwise, then the output consists of a longest common subsequence (LCS) of the input strings. Other notable variants are the minimum sum of pair and the multiple alignment under a fixed evolutionary tree, both to be discussed below. The reader is referred to [8, 18, 94] for comprehensive surveys of the available methods. Moreover, it should also be pointed out that the current methods may not yield biologically adequate alignments [59, 88].

The problem in its general form gives rise to a recurrence that is computed thru an obvious generalization of the dynamic programming algorithm that handles two strings. This produces an algorithm taking time $O(2^k n^k)$ and space $O(n^k)$ to process k strings of comparable length. This performance may be regarded as exponential or polynomial depending on whether k is assumed to be a parameter or a constant. Either way, the computations demanded by this and other methods of multiple sequence comparisons are impractical even for modestly sized sets of relatively short sequences (say, 5-10 sequences each about 100 symbol long). Some versions of the multiple sequence alignment problem were motivated precisely by the biologists' need to settle for suboptimal, biologically plausible solutions, in exchange for computational affordability.

Exact solutions for the LCS are given in [41], [39], [40], [42]. Their typical form is exponential in the number k of strings. In some cases, such a complexity does not even degrade gracefully to the corresponding complexities for the case of 2 strings. While simple greedy algorithms achieve a performance of $Opt + O(Opt^{1/\sqrt{2}})$ for the SCS, and $Opt - O(Opt^{1/2+\epsilon})$ for any $\epsilon > 0$ for the LCS,

the existence of acceptable approximation algorithms can be confuted [43]. The general multiple alignment problem inherits NP-completeness from its LCS and SCS specializations. In fact, NP-completeness results relative to LCS and SCS problems were first achieved by Maier [63]. More recent results are in [89] and [15].

7.1 Minimum Sum of Pairs

For this version of the multiple sequence alignment problem, the cost function d in (4) is given by:

$$d(a_{1j}a_{2j}\dots a_{kj}) = \sum_{1 \leq f < g \leq k} c(a_{fj}, a_{gj}) \quad (5)$$

where $c(\lambda, \lambda) = 0$, $c(a, b)$ is an entry in one of the substitution matrices discussed in Section 4.1 when both a and b are in the alphabet Σ . When only one of the arguments is λ , c is a gap cost (see Section 4.4). Moreover, c must satisfy the triangle inequality.

Intuitively, an optimal multiple alignment in this case is one that best accomodates the pairwise alignment of the strings involved subject to the constraint that all strings must be aligned. For the case $k = 2$, it reduces to standard two-sequence alignment discussed in previous Sections. Notice that Minimum Sum of Pairs alignments (*SP* alignments, for short) tend to maximize the number of positions at which all, or nearly all, of the aligned strings agree. Because of this property, the method is useful in multiple alignments of biological sequences, e.g., proteins, where we want to measure the “variability” of the sequences rather than identify plausible ancestors [6, 16].

In principle, it is not difficult to set up a solution in $O(k^2 2^k n^k)$ time and $O(n^k)$ space. However, *SP* alignment is NP-complete [92]. We point out that NP-completeness of this problem cannot be directly inferred from the NP-completeness of the general alignment problem given by (4), since the cost function d is now specified. Even before the computational complexity of *SP* was settled, there have been two lines of research aiming at the design of fast algorithms for the problem. A common feature of both efforts is to find a good matrix A . That is, a matrix A that, although not optimal, has a cost “close” to the optimum. That is done by using an observation, which we now state.

Let A be any *SP* alignment and let $A_{i,j}$ be the pairwise alignment of string x_i and x_j induced by A . That is, $A_{i,j}$ is obtained from A by deleting all of its rows, except the i -th and the j -th, and then by deleting all columns that have only λ 's. Now, if there is a multiple alignment A such that $A_{i,j}$ gives the optimal pairwise alignment of x_i with x_j , for each $1 \leq i < j \leq k$, then A is an optimal *SP* alignment.

This observation suggests to find a good alignment using the following “idea”. Compute the $\binom{k}{2}$ optimal pairwise alignments of the k strings. By suitably combining those pairwise alignments, build an alignment A in such a way that the cost of its induced pairwise alignments $A_{i,j}$ is “close” to the cost of the optimal pairwise alignment of x_i and x_j , for all $1 \leq i < j \leq n$.

Unfortunately, it is not always possible to combine all pairwise alignments into a multiple alignment A . That is due to the fact that some pairwise alignments may be incompatible for such a combination. So, one usually resorts to the less ambitious task of choosing a subset of pairwise alignments (usually $k-1$) that are compatible, i.e., they can be combined into a multiple alignment. For instance, in the Feng and Doolittle alignment algorithm [25]), one chooses a spanning tree from the complete graph of all pairwise alignments. Now, let's see how the above observations are used.

Carrillo and Lipman have proposed a method that “cuts down” the search space in which the optimal solution lies [16]. Indeed, they give a method to compute upper bounds on the cost of pairwise alignments induced by the optimal alignment we are seeking. Then, the reduction of the search space is achieved by limiting the search to alignment matrices that induce pairwise alignments of cost within the upper bounds. Essential for the determination of those bounds is to find a good,

although not optimal, multiple alignment A . Significant reductions of the computational effort have been reported [61].

Another line of research tries to devise approximation algorithms that guarantee a provably good solution. That is, they compute a matrix A , again not necessarily optimal, but for which one can prove “a priori” how close it is to the optimal solution. The best performance ratio currently available is $2 - 1/k$, for any fixed l . It has been obtained by Bafna and Pevzner [12] improving on earlier remarkable work by Pevzner [73] and Gusfield [33]. From the practical point of view, it seems that the method by Gusfield is the one that, so far, yields biologically plausible solutions that are only 2% from optimal [73].

At the hearth of those approximation algorithms there is a careful investigation of the connection between the compatibility of pairwise alignments to form multiple alignment of k strings and the choice of which pairwise alignments to use. Indeed, first of all, the notion of compatibility has been extended from pairwise alignment to multiple alignments involving l strings, $l < k$. That is, the optimal alignments from which we try to build A consist of l strings each (so, now we have $\binom{k}{l}$ possible alignments to choose from). Moreover, that notion has been formalized in terms of a graph having some specific topological constraints (those graphs are called configurations in [73]). Finally, the problem of finding a good multiple alignment A has been reduced to the one of finding an optimal configuration satisfying some additional topological constraints. This latter task can be done in polynomial time. We remark that not all of the $\binom{k}{l}$ possible alignments of l strings are used to obtain A .

7.2 Traces

As we have discussed in the previous Subsection, even if we have all $\binom{k}{2}$ possible pairwise alignments, only a limited number is used to build the “approximate solution” A . In view of the observation in the previous Subsection, there is a fundamental question that needs to be asked: Given a set of pairwise alignments, how can we find a multiple alignment that is as close as possible to all pairwise alignments in the set? This question was brought to light by Kececioğlu [52], who formalized it and gave algorithms solving it. We briefly review this multiple alignment method.

Given k strings x_1, x_2, \dots, x_k , fix a set F of pairwise alignments of those strings. F can be represented by an alignment graph $G = (V, E)$ together with a partial order relation $<$. V is given by pairs $(i, x_j[i])$, for $1 \leq j \leq k$. There is an edge between two vertices if and only if the two characters in the vertices are matched in a pairwise alignment in F . Moreover, given two vertices v and w , $v < w$ if and only if the character in v immediately precedes the character in w in one of the strings. This partial order relation captures the ordering of characters within a string.

One can observe the following. A path in an alignment graph is a set of characters that can be potentially aligned into a column of a multiple alignment matrix A . Generalizing, the connected components of a set of edges in G can be seen as columns in a multiple alignment A . Those columns are valid if they can be ordered so as to respect the $<$ relation character by character (on the rows of A).

Informally, the maximum trace problem consists of identifying an optimal subset of the edges of E such that their connected components can be combined into a valid multiple alignment A . Optimality is defined so as to “maximize” the use of the information available from the pairwise alignments in F . This problem can be formally stated so as to satisfy formulation (4). Moreover, it contains Minimum Sum of Pairs as a special case [52]. Kececioğlu showed that the general problem is NP -complete and he devised a branch and bound solution for it.

7.3 Multiple Sequence Alignment Under an Evolutionary Tree

Fix a set X of k strings and let Y be a set of hypothetical ancestral strings. An evolutionary tree $T_{X,Y}$ for X is a weighted tree of $|X| + |Y|$ nodes, where the leaves are associated to the strings in X and the internal nodes to the ones in Y . The weight or cost of an edge is usually taken to be the edit distance between the strings at its endpoints, i.e., the cost of the pairwise alignment. The cost of a tree is the sum of the costs of its edges. In its most general form, multiple sequence alignment under an evolutionary tree consists of finding a minimal cost tree $T_{X,Y}$ for X . Minimization is carried out over all sets Y and trees $T_{X,Y}$. The optimal multiple alignment matrix A can be recovered from the optimal tree since any tree of pairwise alignment can be merged into a multiple alignment that complies with the alignments in the tree. An important special case, due to Sankoff [77] and which was the first formulation of this kind of alignment, is for a fixed tree. That is, the topology of the tree is fixed and we must optimize over the strings in Y only. Following Sankoff [77], this “family” of multiple sequence alignments can be cast in the form given by (4). We also point out that this type of alignment tries to minimize the number of mutations from the species represented by strings in X to their ancestors, represented by the strings in Y .

For this class of problems, there are a set of negative results stating that the existence of approximation algorithms that can get arbitrarily close to the optimum is unlikely, even if the topology of the tree is given [92].

Without any tree being assumed, an approximation with a performance ratio of $2 - 2/k$ in $O(k^2n^2)$ time has been established by Gusfield [33]. When the tree is fixed, Sankoff [77] provided an exact algorithm. Studies that generalize to this problem some of the ideas of Carrillo and Lipman for SP alignment are reported in Altschul and Lipman [6] and Pevzner [73]. An approximation with performance ratio of 2 for fixed tree with triangle inequality is afforded in time $O(k^2n^2 + k^3)$ [44].

7.4 Dot-Plots

Multiple sequence comparisons may also be based on measures other than edit scripts, e.g., dot-plots [20]. In its simplest version, the dot-plot associated with a pair of strings is the raw binary matrix of matches between those two strings, i.e., entry $(i, j) = 1$ iff the i th symbol of the first string is identical to the j th symbol of the second string. Criteria more general than mere matching can be adopted, e.g., weighted matches. In the most striking cases, just glancing at a binary dot-plot one finds regions of high similarity in the underlying strings: such regions are in fact overwhelmingly exposed thru visual impact from dense dot areas clustered along diagonals. Working with a set of strings, the issue becomes one of “merging” multiple pairwise dot-plots into a single, meaningful region of density in the hyperspace determined by that set. A natural problem in this context is that of how to define consistency, i.e., the degree of significance which could be assigned to a dot in one of the planes given the existence of the other dot-plots. Dot-plots cannot be simply superimposed, because of deletions and insertions in the various alignment pairs. The alternative proposed in [67] consists of filtering the individual original plots according to a consistency criterion that results in an easy and elegant algebraic computation. Specifically, assume 3 strings are given and let $A_{(1,2)}$, $A_{(1,3)}$ and $A_{(2,3)}$ be the pairwise plots, with obvious meaning. The criterion consists of requiring that a similarity (i, j) in $A_{(1,3)}$ be supported by the appearance, for some k , of dots at (i, k) and (k, j) in $A_{(1,2)}$ and $A_{(2,3)}$, respectively. This validation of dots translates thus to a standard boolean matrix multiplication and can be performed accordingly. The result is a filtered dot-matrix which is now consistent with the other two, according to this criterion. Filtration of the triple is obtained by subjecting the other two plots to the same treatment. For many strings,

the problem reduces to iteration over sets of 3 strings.

In [90], the problem is modeled as a multipartite graph, and “having a similarity in common” is interpreted in terms of a suitably introduced notion of consistency on multipartite graphs. Finding all there is in common between the dot-plots becomes then the problem of finding the maximum consistent subgraph to a given multipartite graph. Algorithms based on this formalization have successfully performed on biological examples.

8 Genome Comparison: A New Class of Sorting Problems

Recall that a genome is a string of DNA. As stated in Section 2, evolution at the level of the genome seems to involve non-local, large scale operations, which can rearrange a whole segment of a chromosome in one evolutionary event. Those non-local operations are: inversion (it replaces a segment of the chromosome with the reverse DNA string); transposition (which moves a segment to a new location of the genome); translocation (it exchanges segments between the ends of two chromosomes); duplication (which copies a segment to a new location); insertion (which inserts a segment) and deletion (which removes a segment). Therefore, new notions of distance have been proposed that quantify the “evolutionary change” of one genome in the other by means of (subsets of) the above operations (see [78] and references therein).

The mechanism through which one can define a distance between genomes is the same used to define any other metric on strings (see Section 2). One formally defines and assigns weights to the operations of interest (say, inversion, transposition and translocation). Then, given two genomes, the distance between the two is defined in exactly the same way as in Section 2, except that the transformation of one genome into the other uses the new set of operations. In what follows, we will assume, for simplicity, that the two genomes we are comparing consist of a single chromosome.

We need some information on how genomes are represented. Most of the current data for genomes is in the form of maps, which provide the location of genes along the chromosomes. For a single chromosome containing n distinct genes, a map can be represented as a permutation of the integers $1, 2, \dots, n$, e.g., $(2, 5, 3, 1)$ is an example of such a map for a “chromosome” with four genes. Sometimes, the genes have an orientation on the map (the reader is referred to [53, 36] for motivation). In that case, each gene is assigned a sign: $+$ or $-$. So, the oriented version of our example “chromosome” could be $(+2, -5, +3, +1)$. For the time being, we will concentrate mainly on maps in which there is no orientation.

Consider two chromosomes we want to compare. When the two maps consist of the same n genes, then the operations of interest are inversion, transposition and translocation. In what follows, we will mainly concentrate on this case and we will define some of those operations formally. We anticipate that all those operations will have unit weights. When the two maps are not composed of the same n genes, then duplication, insertion and deletion are also relevant. We will not consider this case.

8.1 Inversion Distance

We are given the maps of two chromosomes, each consisting of some permutation of the same n genes. That is, the two maps are given by the permutations $\sigma = (\sigma_1, \sigma_2, \dots, \sigma_n)$ and $\tau = (\tau_1, \tau_2, \dots, \tau_n)$. We model the inversion operation described earlier by means of the reversal of the interval $[i, j]$. Formally, a reversal of the interval $[i, j]$ is the permutation $\rho = i, i+1, \dots, j \rightarrow j, j-1, \dots, i$. Applying ρ to σ by the composition $\sigma \cdot \rho$ has the effect of reversing the order of the genes $\sigma_i, \dots, \sigma_j$. For instance, let $\sigma = (2, 4, 3, 1)$. A reversal $[2, 3]$ applied to σ would yield the new permutation $(2, 3, 4, 1)$.

We are interested in the following problem. Given the permutations σ and τ , find a series of reversals $\rho_1, \rho_2, \dots, \rho_d$ such that $\sigma \cdot \rho_1 \cdot \rho_2 \cdots \rho_d = \tau$, where d is minimum. d is referred to as the reversal distance. Notice that problem reduces to finding the reversal distance between the permutation $\phi = \tau^{-1}\sigma$ and the identity permutation $\iota = (1, 2, \dots, n)$, where τ^{-1} be the inverse of τ . We will work with this version of the problem, which is known as sorting by reversal. Since reversals generate the symmetric group S_n , an alternative statement of the problem is: given an arbitrary permutation ϕ of S_n , find the shortest product of generators that equals ϕ . Let $d(n) = \max_{\phi \in S_n} d(\phi)$, where $d(\phi)$ is the reversal distance of ϕ from the identity permutation.

Sorting by reversal was posed by [96]. Those authors also devised an heuristic algorithm computing it. Very recently, Kececioglu and Sankoff [54] started a formal study of the computational complexity of this problem and conjectured that it is NP-Hard. The relationship between sorting by reversal and other fairly well understood problems in Computer Science that might help settle this question is discussed in [54].

Kececioglu and Sankoff also devised two algorithms. One is a polynomial time approximation algorithm that finds a solution within a factor of two from optimum. It is the first algorithm for which one can show a performance guarantee bound on the solution. The other one is a branch and bound algorithm that always finds an exact solution to the problem. In order to have a fast convergence of the branch and bound search, one needs to come up with nontrivial lower and upper bounds on the reversal distance d . Both algorithms have been shown to perform well on biologically important data [54].

Bafna and Pevzner [14] have recently obtained a $7/4$ approximation algorithm. In order to obtain that result, they had to study some combinatorial and probabilistic problems on S_n that are interesting in their own right and relevant to molecular biology as well. They proved that $d(n) = n - 1$, settling a conjecture by Gollan (see [14, 54]). They also studied the problem of expected reversal distance between random permutations and showed that it is very close to $d(n)$. That is an indication of the fact that reversal distance provides a good separation between related and non-related sequences in molecular evolution studies.

A related problem is the one of sorting signed permutations by reversals. ϕ is still a permutation on $\{1, 2, \dots, n\}$, but each element of ϕ has a $+$ or $-$ sign, e.g., $(+1, -5, +4, -3, +2)$ is a signed permutation. Now, a reversal $[i, j]$ is defined as above, but it changes also the signs of the elements $\phi_i, \phi_{i+1}, \dots, \phi_j$. For instance, $[3, 4]$ applied to $(+1, -5, +4, -3, +2)$ yields $(+1, -5, +3, -4, +2)$. We are interested in finding the shortest sequence of reversals that transforms ϕ into the identity signed permutation $\iota = (+1, +2, \dots, +n)$. Motivation for the biological relevance of this problem can be found in [36, 53]. We limit ourselves to mention that an optimal solution to the signed case gives an excellent approximation of the reversal distance for the case of unsigned permutations presented earlier [36, 53].

As in the unsigned case, approximation algorithms were developed first (see [14, 53]). Surprisingly, the signed case can be solved in polynomial time [36].

8.2 Some Other Notions of Distance

Another operation for which a distance has been defined is transposition. Its relevance to biology is addressed in [35]. Again, the two genomes are (unsigned) permutations σ and τ of the integers in $[1, n]$. A transposition $\rho(i, j, k)$ exchanges the "place of the integers" $[i, j - 1]$ and $[j, k - 1]$ in the interval $[i, k - 1]$. For instance $\rho(2, 4, 6)$ transforms $(2, 3, 4, 5)$ into $(4, 5, 2, 3)$. $\sigma \cdot \rho(i, j, k)$ has the effect of exchanging $\sigma_i, \dots, \sigma_{j-1}$ with $\sigma_j, \dots, \sigma_{k-1}$. The transposition distance between σ and τ can be defined in the same way as the inversion distance between those two permutations, except that we apply transposition transformations rather than reversals. Again, since transpositions

generate the symmetric group S_n , the transposition distance between σ and τ can be reduced to the computation of the transposition distance between $\phi = \tau^{-1}\sigma$ and the identity permutation. This problem is referred to as sorting by transposition. Again, as in the case of sorting by reversal, it has been conjectured that it is NP-Hard. The first provably good approximation algorithm has been obtained in [13].

The distances that we have defined so far use only one kind of operation, i.e., either reversal or transposition. Kececioglu and Ravi [55] have developed a model for translocation that allows them to define a distance in terms of that operation. They also defined a distance using two operations: inversion and translocation. The computational complexity of those new distances is also an open problem. For the time being, approximation algorithms are available [55].

Much work remains to be done in this area. For instance, we have considered some distance problems in which the involved operations have unit weight. It would be interesting to consider the case in which those operation are assigned a weight (restricted to reversal distance, this problem has been proposed by [54]). Such an investigation would lead to two new areas of research: How to assign weights to the chosen operations (for a brief discussion of this topic see [78]) and how to compute those distances.

9 Acknowledgements

This paper is based on presentations and literature given at the Workshop on "Sequence Alignment" which was held in Princeton, N.J., Nov 10-12, 1995, as part of the DIMACS Special Year on Mathematical Support for Molecular Biology. The authors gratefully acknowledge the support and dedication of their colleagues in the Workshop Committee, most of whom also served as lecturers delivering the notes that the present paper tried to capture: S. Altschul, D. Brutlag, R. Doolittle, M. Farach, P. Green, D. Gusfield, S. Henikoff, J. Kececioglu, D. Lipman, W. Miller, W. Pearson, P. Pevzner, D. Sankoff, J. Schmidt, G. Stormo, M. Vingron, M. Waterman. Thanks are also expressed to DIMACS and to the Steering Committee of its 1994-95 Special Year for devising this topic, stimulating and supporting every aspect of that meeting.

References

- [1] A. Aggarwal and J. Park. Searching in multidimensional monotone matrices. In *Proc. 29th Symposium on Foundations of Computer Science*, pages 497-512. IEEE, 1988.
- [2] A. Lempel and J. Ziv. On the complexity of finite sequences. *IEEE Trans. on information Theory*, 22:75-81, 1976.
- [3] S.F. Altschul. Amino acid substitution matrices from an information theoretic perspective. *J. Mol. Bio.*, 219:555-565, 1991.
- [4] S.F. Altschul. A protein alignment scoring system sensitive to all evolutionary distances. *J. Mol. Evol.*, 36:290-300, 1993.
- [5] S.F. Altschul, W. Gish, W. Miller, E.W. Myers, and D.J. Lipman. Basic local alignment search tool. *J. Mol. Bio.*, 215:403-410, 1990.
- [6] S.F. Altschul and D.J. Lipman. Trees, stars, and multiple biological sequence alignment. *SIAM J. Appl. Math.*, 49(1):197-209, 1989.
- [7] A. Apostolico, M.J. Atallah, L.L. Larmore, and S. McFadin. Efficient parallel algorithms for string editing and related problems. *Siam J. on Computing*, 19:968-988, 1990.
- [8] P. Argos, M. Vingron, and G. Vogt. Protein sequence comparison: Methods and significance. *Protein Engineering*, 4:375-383, 1991.

- [9] R. Arratia, L. Gordon, and M.S. Waterman. An extreme value theory for sequence matching. *Ann. Stat.*, 14:971-993, 1986.
- [10] R. Arratia and M.S. Waterman. The Erdos-Renyi strong law for pattern matching with a given proportion of mismatches. *Ann. Prob.*, 17:1152-1169, 1989.
- [11] V. Bafna, E. Lawler, and P.A. Pevzner. Approximation algorithms for multiple sequence alignment. In A. Apostolico, M. Crochemore, Z. Galil, and U. Manber, editors, *Proc. Combinatorial Pattern Matching 94 (Lecture Notes in Computer Science, Vol. 807)*, pages 43-53, Berlin, 1994. Springer-Verlag.
- [12] V. Bafna and P. Pevzner. Approximate methods for sequence alignment. *manuscript*, 1993.
- [13] V. Bafna and P. Pevzner. Sorting permutations by transpositions. In *Proc. 6th Symposium on Discrete Algorithms*, pages 614-621. ACM-SIAM, 1995.
- [14] V. Bafna and P. Pevzner. Genome rearrangments and sorting by reversals. *Siam J. on Computing.*, 25, 1996.
- [15] H.L. Bodlaender and M.R. Fellows and M.T. Hallett. Beyond np-completeness for problems of bounded width: Hardness for the w hierarchy (extended abstract). *ACM Sympos. Theory Comput.*, 26:449-458, 1994.
- [16] H. Carrillo and D. Lipman. The multiple sequence alignment problem in biology. *SIAM J. Appl. Math.*, 48(5):1073-1082, 1988.
- [17] L. Carrol. A new puzzle. *Vanity Fair*, 1879.
- [18] S.C. Chan, A.K.C. Wong, and D.K.Y. Chui. A survey of multiple sequence comparison methods. *Bull. Math. Biol.*, 54:563-598, 1992.
- [19] V. Chvatal and D. Sankoff. Longest common subsequence of two random sequences. *J. of Appl. Probab.*, 12:306-315, 1975.
- [20] J.F. Collins and A.F.W. Coulson. Molecular sequence comparison and alignment. pages 323-358, 1987.
- [21] M. O. Dayhoff and R. V. Eck. *Atlas of Protein Sequences and Structures*. National Biomedical Res. Foundation, 1968.
- [22] M.O. Dayhoff, R.M. Schwartz, and B.C. Orcutt. A model of evolutionary change in proteins. In M.O. Dayhoff, editor, *Atlas of Protein Sequence and Structure*, pages 345-352. Nat. Biomed. Res. Found., 5, supp. 3, 1978.
- [23] T. Dobzhanski and A.H. Sturtvant. Inversions in the chromosomes of drosophila pseudoscura. *Genetics*, 23:28-64, 1938.
- [24] R.F. Doolittle. Similar amino acid sequences: Chance or common ancestry? *Science*, 214:149-159, 1981.
- [25] D. Feng and R. Doolittle. Progressive sequence alignment as a prerequisite to correct phylogenetic trees. *Journal of Molecular Evolution*, 25:351-360, 1987.
- [26] D.F. Feng, M.S. Johnson, and R.F. Doolittle. Aligning amino acid sequences: Comparison of commonly used methods. *J. Mol. Evol.*, 21:112-125, 1985.
- [27] W.M. Fitch and E. Margoliash. Construction of phylogenetic trees. *Science*, 155:279-284, 1967.
- [28] R.G. Gallager. *Information Theory and Reliable Communication*. John Wiley and Sons, 1968.
- [29] L.L. Gatlin. The information content of DNA. *J. Theor. Biol.*, 10:281-300, 1966.
- [30] R. Giancarlo. Dynamic programming-special cases. In A. Apostolico and Z. Galil, editors, *Pattern Matching Algorithms*, New York, to appear. Oxford University Press.
- [31] O. Gotoh. An improved algorithm for matching of biological sequences. *Journal of Molecular Biology*, 162:705-708, 1982.

- [32] A.M. Griffin and M.E.G. Boursnell. Analysis of the nucleotide sequence of DNA from the region of the thymidine kinase gene of infectious laryngotracheitis virus; potential evolutionary relationships between the herpes virus subfamilies. *J. Gen. Virol.*, 71:841–850, 1990.
- [33] D. Gusfield. Efficient methods for multiple sequence alignment with guaranteed error bounds. *Bull Math. Biol.*, 55(1):141–154, 1993.
- [34] R. W. Hamming. Error detecting and error correcting codes. *Bell System Tech. J.*, 29:147–160, 1950.
- [35] S. Hannenhalli, C. Chappey, E. Koonin, and P. Pevzner. Algorithms for genome rearrangements: Herpesvirus evolution as a test case. In *3rd Int. Conference on Bioinformatics and Complex Genome Analysis*, 1994.
- [36] S. Hannenhalli and P. Pevzner. Transforming cabbage into turnip (polynomial algorithm for sorting signed permutations by reversals). In *Proc. 27th Symposium on Theory of Computing*, pages 178–187. ACM, 1995.
- [37] S. Henikoff and J. Henikoff. Amino acid substitution matrices from protein blocks. *Proc. Nat. Acad. of Sci., USA*, 89:10915–10919, 1992.
- [38] S. Henikoff and J. Henikoff. Performance evaluation of amino acid substitution matrices. *Proteins: Structure, function and genetics*, 17:49–61, 1993.
- [39] W.J. Hsu and Du M.W. Computing a longest common subsequence for a set of strings. *BIT*, 24:45–59, 1984.
- [40] R.W. Irving and C.B. Fraser. Two algorithms for the longest common subsequence of three (or more) strings. In *Third Annual Symposium, CPM 92, Tucson, Arizona, April 29 - May 1, 1992. Proceedings. (Lecture Notes in Computer Science, Vol. 644)*, pages 214–229, 1992.
- [41] S. J. Itoga. The string merging problem. *BIT*, 21:20–30, 1981.
- [42] G. Jacobson and K.P. Vo. Heaviest increasing/common subsequence problems. In *Third Annual Symposium, CPM 92, Tucson, Arizona, April 29 - May 1, 1992. Proceedings. (Lecture Notes in Computer Science, Vol. 644)*, pages 52–66, 1992.
- [43] Jiang and M. Li. Optimization problems in molecular biology. In D.Z. Du and J. Sun, editors, *Advances in Optimization and Approximation (Manuscript received 11 February 1994)*, 1993.
- [44] T. Jiang, E.L. Lawler, and L. Wang. Aligning sequences via an evolutionary tree: Complexity and approximation. *ACM Sympos. Theory Comput.*, 26:760–760, 1994.
- [45] M.I. Kanehisa and W.B. Goad. Pattern recognition in nucleic acid sequences I: A general method for finding local homologies and simmetries. *Nucl. Acid Res.*, 10:247–264, 1982.
- [46] S.K. Kannan and E.W. Myers. An algorithm for locating non-overlapping regions of maximum alignment score. In A. Apostolico, M. Crochemore, Z. Galil, and U. Manber, editors, *Proc. Combinatorial Pattern Matching 94 (Lecture Notes in Computer Science, Vol. 684)*, pages 74–86, Berlin, 1993. Springer-Verlag.
- [47] S. Karlin and S.F. Altschul. Methods for assessing the statistical significance of molecular sequence features by using general scoring systems. In *Proc. Nat. Acad. of Sci., U.S.A.*, volume 87, pages 2264–2268, 1990.
- [48] S. Karlin, A. Dembo, and T. Kawabata. Statistical significance of high scoring segments from molecular sequences. *Ann. Stat.*, 18:571–581, 1990.
- [49] S. Karlin, G. Glandour, F. Ost, S. Tavare, and L.J Korn. New approaches for computer analysis of nucleic acid sequences. *Proc Nat Acad Sci USA*, 80:5660–5664, 1983.
- [50] S. Karlin, E.S. Mocarski, and G.A. Schachtel. Molecular evolution of herpesviruses: genomic and protein sequence comparison. *J. Virol.*, 68:1886–1902, 1994.
- [51] S. Karlin and F. Ost. Maximum length of common words among random letter sequences. *Ann. Prob.*, 16:535–563, 1988.

- [52] J. Kececioglu. The maximum weight trace problem in multiple sequence alignment. In *4th Annual Symposium, CPM 93, Padova, Italy, June 2-4, 1993. Proceedings. (Lecture Notes in Computer Science)*, volume 684, pages 106–119, 1993.
- [53] J. Kececioglu and D. Sankoff. Efficient bounds for oriented chromosome inversion distance. In A. Apostolico, M. Crochemore, Z. Galil, and U. Manber, editors, *Proc. Combinatorial Pattern Matching 94 (Lecture Notes in Computer Science, Vol. 807)*, pages 307–325, Berlin, 1994. Springer-Verlag.
- [54] J. Kececioglu and D. Sankoff. Exact and approximation algorithms for sorting by reversals, with application to genome rearrangement. *Algorithmica*, 13:180–210, 1995.
- [55] J.D. Kececioglu and R. Ravi. Of Mice and Men: Algorithms for evolutionary distances between genomes with translocation. In *Proc. 6th Symposium on Discrete Algorithms*, pages 604–613. ACM-SIAM, 1995.
- [56] A. N. Kolmogorov. Three approaches to the quantitative definition of information. *Problemi Pederachi Inf.*, 1, 1965.
- [57] G.M. Landau and J.P. Schmidt. An algorithm for locating tandem repeats. In A. Apostolico, M. Crochemore, Z. Galil, and U. Manber, editors, *Proc. Combinatorial Pattern Matching 94 (Lecture Notes in Computer Science, Vol. 684)*, pages 120–133, Berlin, 1993. Springer-Verlag.
- [58] E. Lander, J.P. Mesirov, and W. Taylor. Study of protein sequence comparison metrics on the Connection Machine CM-2. *J. Supercomputing*, 3:255–269, 1989.
- [59] A.M. Lesk, M. Levitt, and C. Chothia. Alignment of amino acid sequences of distantly related proteins using variable gap penalties. *Protein Engineering*, 1:77–78, 1986.
- [60] V.I. Levenshtein. Binary codes capable of correcting deletions, insertions and reversals. *Soviet Phys. Dokl.*, 6:707–710, 1966.
- [61] D.J. Lipman, S.F. Altschul, and J.D. Kececioglu. A tool for multiple sequence alignment. *Proc. Nat. Acad. Sci. (USA)*, 86:4412–4415, 1989.
- [62] D.J. Lipman and W.L. Pearson. Rapid and sensitive protein similarity searches. *Science*, 2:1435–1441, 1985.
- [63] D. Maier. The complexity of some problems on subsequences and supersequences. *J. Assoc. Comput. Mach.*, 25(2):322–336, 1978.
- [64] P. Martin-Lof. The definition of random sequences. *Information and Control*, 9, 1966.
- [65] A.D. McLachlan. Tests for comparing related amino acid sequences. Cytochrome c and cytochrome c551. *J. Mol. Biol.*, 61:409–424, 1971.
- [66] W. Miller. An algorithm for locating a repeated region. *manuscript*, 1992.
- [67] M. Vingron and P. Argos. Motif recognition and alignment for many sequences by comparison of dot-matrices. *J. Mol. Biol.*, 218:33–43, 1991.
- [68] S.B. Needleman and C.D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *J. Mol. Biol.*, 48:433–443, 1970.
- [69] J. Palmer and L. Herbon. Plant mitochondrial DNA evolves rapidly in structure, but slowly in sequence. *J. Mol. Evol.*, 27:87–97, 1988.
- [70] S. Pascarella and P. Argos. Analysis of insertion/deletion in proteins. *J. Mol. Biol.*, 224:461–471, 1992.
- [71] W.L. Pearson. Searching protein sequence libraries: Comparison of the sensitivity and selectivity of the Smith-Waterman and FASTA algorithms. *Genomics*, 11:635–650, 1991.
- [72] W.L. Pearson and W.R. Lipman. Improved tools for biological sequence comparison. *Proc. Nat. Acad. Sci., USA*, 85:2444–2448, 1988.
- [73] P. Pevzner. Multiple alignment, communication cost, and graph matching. *Siam J. on Applied Math.*, 52:1763–1779, 1992.

- [74] K.R. Popper. *The Logic of Scientific Discovery*. Hutchinson, London, 1959.
- [75] J.K. Mohana Rao. New scoring matrix for amino acid residue exchanges based on residue characteristic physical parameters. *Int. J. Pept. Protein Res.*, 29:276–281, 1987.
- [76] J.L. Risler, M.O. Delorme, H. Delacroix, and A. Henaut. Amino acid substitutions in structurally related proteins. a pattern recognition approach. Determination of a new and efficient scoring matrix. *J. Mol. Biol.*, 204:1019–1029, 1988.
- [77] D. Sankoff. Minimal mutation trees of sequences. *SIAM J. Appl. Math.*, 28(1):35–42, 1975.
- [78] D. Sankoff. Edit distance for genome comparison based on non-local operations. In A. Apostolico, M. Crochemore, Z. Galil, and U. Manber, editors, *Proc. Combinatorial Pattern Matching 92 (Lecture Notes in Computer Science, Vol. 644)*, pages 121–135, Berlin, 1992. Springer-Verlag.
- [79] D. Sankoff and J.B. Kruskal editors. *Time Warps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison*. Addison-Wesley, Reading, MA, U.S.A., 1983.
- [80] D. Sankoff, G. Leduc, N. Antoine, B. Paquin, B. Franz Lang, and R. Cedergren. Gene order comparison for phylogenetic inference: evolution of the mitochondrial genome. *Proc. Nat. Acad. Sci., USA*, 89:6575–6579, 1992.
- [81] J.P. Schmidt. All highest scoring pairs in weighted grid graphs and its application to finding all approximate repeats in strings. *manuscript*, 1995.
- [82] P.H. Sellers. The theory and computation of evolutionary distances: Pattern recognition. *J. of Alg.*, 1:359–373, 1980.
- [83] P.H. Sellers. Pattern recognition in genetic sequences by mismatch density. *Bull. Math. Biol.*, 46:501–514, 1984.
- [84] R.F. Smith and T.F. Smith. Automatic generation of primary sequence patterns from sets of related protein sequences. *Proc. Nat. Acad. of Sci., USA*, 87:118–122, 1990.
- [85] T. Smith and M.S. Waterman. Identification of common molecular subsequences. *J. Mol. Biol.*, 147:195–197, 1981.
- [86] T. Smith, M.S. Waterman, and Burks. The statistical distribution of nucleic acid similarities. *Nucleic Acid Research*, 13:645–656, 1985.
- [87] D.J. States, W. Gish, and S.F. Altschul. Improved sensitivity of nucleic acid databases searches using application specific scoring matrices. *METHODS: A companion to Method. in Enzim.*, 3:66–70, 1991.
- [88] W.R. Taylor. The classification of amino acid conservation. *J. of Theor. Biol.*, 119:205–218, 1986.
- [89] V.G. Timkovskii. Complexity of common subsequence and supersequence problems and related problems. *Cybernetics*, 25(5):565–580, 1990.
- [90] M. Vingron and P. Pevzner. Multiple sequence comparison and consistency on multipartite graphs. *manuscript*, 1994.
- [91] R. von Mises. *Probability, Statistics and Truth*. MacMillan, New York, 1939.
- [92] L. Wang and T. Jiang. On the complexity of multiple sequence alignment. *Journal of Computational Biology, to appear*, 1993.
- [93] S. Watanabe. *Knowing and Guessing*. Wiley, New York, 1969.
- [94] M.S. Waterman. Sequence alignments. In M.S. Waterman, editor, *Mathematical Analysis of DNA Sequences*, pages 53–92. CRC, 1989.
- [95] M.S. Waterman, L. Gordan, and R. Arratia. Phase transition in sequence matches and nucleic acid structure. *Proc. Natl. Acad. Sci., USA*, 84:1239–1243, 1987.
- [96] G.A. Wattenson, W.J. Ewens, T.E. Hall, and A. Morgan. The chromosome inversion problem. *J. of Theor. Biol.*, 99:1–7, 1982.

- [97] W.J. Wilbur and D. J. Lipman. The context dependent comparison of biological sequences. *SIAM J. Appl. Math.*, 44:557-567, 1984.
- [98] W.J. Wilbur and D.J. Lipman. Rapid similarity searches of nucleic acid and protein data banks. In *Proc. Nat. Acad. Sci. USA 80*, pages 726-730, 1983.