An Overview of Sequence Comparison: Time Warps, String Edits, and Macromolecules
Author(s): Joseph B. Kruskal
Source: *SIAM Review*, Vol. 25, No. 2 (Apr., 1983), pp. 201-237
Published by: Society for Industrial and Applied Mathematics
Stable URL: http://www.jstor.org/stable/2030214
Accessed: 25-11-2017 15:20 UTC

JSTOR is a not-for-profit service that helps scholars, researchers, and students discover, use, and build upon a wide range of content in a trusted digital archive. We use information technology and tools to increase productivity and facilitate new forms of scholarship. For more information about JSTOR, please contact support@jstor.org.

Your use of the JSTOR archive indicates your acceptance of the Terms & Conditions of Use, available at http://about.jstor.org/terms

Society for Industrial and Applied Mathematics is collaborating with JSTOR to digitize, preserve and extend access to *SIAM Review*

This content downloaded from 101.78.168.41 on Sat, 25 Nov 2017 15:20:11 UTC
All use subject to http://about.jstor.org/terms

SIAM Review
Vol. 25, No. 2, April 1983

© 1983 Society for Industrial and Applied Mathematics
0036-1445/83/2502-0003 $01.25/0

AN OVERVIEW OF SEQUENCE COMPARISON: TIME WARPS, STRING EDITS, AND MACROMOLECULES*

JOSEPH B. KRUSKAL†

Abstract. A wide variety of different applications lead to problems in which sequences of different lengths must be compared, to see how different they are, and to see which elements in one sequence correspond to which elements in the other sequence. Successful methods for handling these problems been repeatedly reinvented which incorporate two basic ideas: a systematic concept of *distance* between sequences, and elegant *recursive algorithms* for doing the necessary computations. The major application areas are speech processing and macromolecular biology. Computer science is another significant application area, and applications have also been made to bird song, handwriting analysis, gas chromatography, geological strata, and text collation. This paper surveys the applications, methods and theory of sequence comparison.

CONTENTS

1. Introduction

It is often necessary to compare two or more sequences, or strings, or vectors, or continuous functions of time, etc. and measure the extent to which they differ. Comparison is used for identification, for error-correction, and for determining relationships. In many situations there is a natural correspondence between the elements, or components, or coordinates, or points of time, etc. in one sequence and those in the other, and the only sensible comparison is between corresponding elements. In such situations it is easy to make the comparison. *Sequence comparison* deals with the more difficult comparisons which arise when the correspondence is not known in advance, perhaps because some underlying correspondence has been disturbed by the gain or loss of elements in one or both sequences.

Most existing methods for comparing sequences to see how different they are can be applied only to sequences of equal length, and are based on comparing corresponding elements. For example, some well-known methods for comparing sequences **a** and **b** are Euclidean distance $\left[\sum_{i=1}^{n} (a_i-b_i)^2\right]^{1/2}$, city block distance

*Received by the editors November 6, 1981, and in final revised form August 2, 1982. This paper is a slightly modified version of the first chapter of *Theory and Practice of Sequence Comparison: Time-Warps, String Edits, and Macromolecules*, David Sankoff and Joseph Kruskal, eds., Addison-Wesley, Reading, Massachusetts, 1983. This paper was typeset by Carmela Patuto at Bell Laboratories, Murray Hill, New Jersey, using the **troff** program running under the Unix^{TM} operating system. Final copy was produced on October 5, 1982.

†Bell Laboratories, Murray Hill, New Jersey 07974.

This content downloaded from 101.78.168.41 on Sat, 25 Nov 2017 15:20:11 UTC
All use subject to http://about.jstor.org/terms

$\sum_{i=1}^{n} |a_i - b_i|$, and Hamming distance, which is simply the number of positions in which the corresponding elements are different. Here a_i corresponds to b_i, and the comparison between them is expressed in the term $(a_i - b_i)^2$ or the term $|a_i - b_i|$ or, in the case of Hamming distance, a term which is 1 if $a_i \neq b_i$ and 0 if $a_i = b_i$.

In some situations the appropriate correspondence to use is not known in advance, either because a_i has no special connection with b_i or simply because the sequences have different lengths. The general approach used in sequence comparison is to seek the appropriate correspondence by optimizing over all possible correspondences which satisfy suitable conditions, such as preserving the order of the elements in the sequence. The optimization procedures used fall within the framework of dynamic programming, though in several cases they were invented without explicit use or apparent awareness of that framework. (Knowledge of dynamic programming will not be required in this paper.) The central themes of sequence comparison are:

(1) distance functions which are appropriate to use when there is no natural correspondence of elements;
(2) optimum correspondences between sequences;
(3) dynamic programming algorithms for calculation;
(4) applications; and
(5) the properties of distances, optimum correspondences, and algorithms.

The applications fall into two major categories: discrete and continuous. In the discrete case, the objects under study are ordinarily sequences (typically of different lengths), though limited attention has also been given to more complicated structures. An individual sequence can be written $\mathbf{a} = (a_1, \cdots, a_n)$, where the elements a_i are drawn from some *alphabet*. The alphabet might consist of all numbers or of all vectors in some low-dimensional vector space, but more often it is a special alphabet characteristic of the application, e.g., the 20 amino acids, or the 4 nucleotides, or the symbols on some particular computer terminal keyboard, or the 40 phonemes of English, or the 113 syllables (properly, "morae") of Japanese. In the continuous case, the objects under study are ordinarily continuous functions of a variable t, usually time, over intervals of different lengths. An individual function can be written $\mathbf{a}(t)$, over the interval 0 to T, where the function values may be numbers, e.g., the amount of material which emerges from a gas chromatograph column at time t, or may lie in a vector space, e.g., a vector of coefficients which describes the frequency content of a speech waveform at time t. The main method for dealing with continuous situations is to convert them into discrete ones, by sampling the time interval at discrete times $t_1, t_2, \cdots$.

In discrete situations, sequence deletions and insertions may result from processes which lead to actual physical deletion and insertion. In situations which have been converted from continuous to discrete by sampling, it is also possible for physical deletion and insertion to occur. In addition, however, compression and expansion along the time axis can give rise to changes which closely resemble deletions and insertions, and which have often been so described. There are, however, subtle differences between the methods used to handle deletion-insertion and compression-expansion.

1a. Applications to molecular biology. Proteins and nucleic acids are macromolecules central to the biochemical activity of all living things, including

This content downloaded from 101.78.168.41 on Sat, 25 Nov 2017 15:20:11 UTC
All use subject to http://about.jstor.org/terms

chemical regulation and genetic determination and transmission. These macromolecules may be considered as long sequences of subunits linked together sequentially in a chain. In the case of proteins, the units are drawn from an alphabet of twenty amino acids, and the sequences are typically a few hundred units long. In the case of nucleic acids, the units are drawn from an alphabet of four different nucleotides (or bases), and the sequences are typically from tens to thousands of units long (for ribonucleic acid, i.e., RNA) or millions of units long (for deoxyribonucleic acid, i.e., DNA).

The methods of sequence comparison are helping to answer several important questions in biology. One set of questions revolves around *homology* of macromolecules: two structures are said to be homologous if they have a common evolutionary ancestor, and are said to have a high degree of homology if the differences which have developed between them are relatively minor. Are certain macromolecules homologous? Which parts of one molecule are homologous to which parts of the other? If we compare many different molecules in one species with the corresponding molecules in another species, how high is the typical degree of homology? For what pairs of species is this typical degree of homology high? Knowledge of homology sheds light not only on evolution, but also on structure and function in organisms. Another set of questions revolves around the structural configuration of individual RNA molecules, i.e., the way in which different parts of the chain join to each other, since the methods of sequence comparison can help determine this also. For access to the substantial literature on the use of sequence comparison in molecular biology, consult such papers as Erickson and Sellers [*][1], Sankoff, Kruskal, Cedergren and Mainville [*], Beyer et al. (1974), and the entire issue of Nucleic Acid Research, volume 10, No. 1, 1982, particularly Good and Kanehisa (1982), Kanehisa and Good (1982), Nussinov et al. (1982a, 1982b), and Auron et al. (1982).

The first known use of dynamic programming methods (though not under that name) for comparing sequences occurred in connection with homology of macromolecules (Needleman and Wunsch, 1970; but see the citation in Barker et al., 1969). Subsequently at least two other groups independently invented similar methods for similar purposes.

1b. Applications to codes and error control. Strings of symbols are used extensively to code information for transfer over a noisy channel such as radio, telegraph, microwave transmission, and so forth. Frequently just two symbols are used, which may be denoted by 0 and 1 regardless of physical implementation. Since noise in the channel introduces error into the received signal, many questions arise about detection and correction of error. The systematic study of different coding schemes and how well they work is referred to as (mathematical) coding theory (see, e.g., MacWilliams and Sloane, 1977), a large field which closely intertwines at some points with information theory.

Though the primary focus of coding theory has always been on substitution errors (in which the wrong symbols are received), it is also true that insertion and deletion errors (in which too many or too few symbols are received) have long been studied under the heading of synchronization error. However, it was only in the 1960's that attention was given to reconstructing the transmitted sequence

[1]Citations identified with [*] are to papers in the book by Sankoff and Kruskal that is described in footnote * on the title page.

This content downloaded from 101.78.168.41 on Sat, 25 Nov 2017 15:20:11 UTC
All use subject to http://about.jstor.org/terms

(i.e., decoding) in the neighborhood of an insertion or deletion error (e.g., F. F. Sellers, 1962; Levenshtein, 1965a, 1965b; Ullman, 1966). Levenshtein (1965a) presented the earliest known use of a distance function which is appropriate in the presence of insertion and deletion errors. His distance function and generalizations of it play a major role in sequence comparison.

Computational methods like those described here seem to be of limited interest in coding theory, probably because a single received sequence (which is only a small part of one message) is virtually never important enough to merit the kind of extended attention which is given to a single sequence in other fields of application. Perhaps as a result, coding theory has not developed these ideas very much, despite Levenshtein's invention of a distance function. See, however, work by F. F. Sellers (1962), Calabi and Hartnett (1969a,b), Levenshtein (1965a,b; 1971), Mills (1978), Tanaka and Kasai (1976), Ullman (1966, 1967), and Kunze and Thierrin (1981).

1c. Applications to computer science. Another context where strings of symbols are of great importance is computer science, since they are so prominent during input and output (where the symbols may be letters, digits, etc.) and during internal functioning (where the symbols may be bits, bytes, words, records, etc). A good review in the field of computer science citing many applications is given by Hall and Dowling (1980), though their interests are less sharply focussed than ours. One practical application in this field, often referred to as *string-correction* or *string-editing*, is the correction of human errors made at the input stage (the keyboard of a computer terminal or a keypunch machine). The earliest known work of this kind is Damerau (1964), Alberga (1967), and Morgan (1970), though these forerunners included neither the distance concept nor the dynamic programming method. Wagner and Fischer (1974) sparked much interest in this problem among computer scientists, and there is now considerable literature. Another application, now in routine use, is the comparison of two computer files, perhaps slightly different versions of the same basic information, where each line or record of the file may be treated as a single symbol. A full description is provided in Hunt and McIlroy (1976), but published material contains only very brief references, as in Kernighan, Lesk, and Ossana (1978) and Johnson (1980). A related application of a more elaborate algorithm is given in Gosling (1981).

One unusual feature has been introduced by Wagner and colleagues in connection with string-correction: the extension to include what are called *swaps* or *transpositions* (i.e., the interchange of adjacent elements in the sequence), a common human error. Their inclusion increases realism, though at the cost of complicating the algorithms and the mathematics.

1d. Applications to human speech. A very active area for sequence comparison methods is speech research, where they are often referred to as *(dynamic) time-warping* or *dynamic programming* or elastic matching. A recent volume of selected reprints, Dixon and Martin (1979), contains many papers explicitly making use of time-warping, and four or five groups in speech processing are known to have independently invented and published dynamic programming methods for this purpose between 1968 and 1973 (Vintsyuk, 1968; Sakoe and Chiba, 1970, 1971; Velichko and Zagoruyko, 1970; Bridle, 1973; and Haton, 1973, 1974a). Dynamic time-warping is now routinely incorporated into most modern systems concerned with recognition of speech or speaker. For

This content downloaded from 101.78.168.41 on Sat, 25 Nov 2017 15:20:11 UTC
All use subject to http://about.jstor.org/terms

further information about applications to recognition of isolated words, see papers such as Itakura (1975), White (1978), Sakoe and Chiba (1978), Myers, Rabiner, and Rosenberg (1980), Rabiner, Rosenberg, and Levinson (1978), and White and Neely (1976). For information about recognition of connected speech, see papers such as Myers and Rabiner (1981a,b), Rabiner and Schmidt (1980), and Bridle and Brown (1979).

Physically, speech is transmitted as a pressure wave through the atmosphere. Conceptually, a preliminary step converts the received pressure wave into a vector function of time, $\mathbf{a}(t)$, where the several-dimensional vector at t describes the pressure wave in a short interval surrounding t. (For example, the coefficients of the vector might give the energy in several frequency bands, or give the "linear predictor coefficients".) Before actual use, however, $\mathbf{a}(t)$ is converted into a sequence of vectors or phonetic symbols, which is the information actually dealt with, by sampling at regular times or by more elaborate means such as dividing the wave into natural segments, possibly followed by classification into phonetic categories.

Consider the much-studied problem of recognizing an isolated word selected from a limited vocabulary. A very common approach is to provide the recognizer with one or several versions of each vocabulary word in sequence form. To recognize an unknown word, the recognizer converts it to sequence form, compares it to many stored sequences, and selects the one which matches best. Of course, a perfect match is unlikely. Compression and expansion may occur, because the rate of speaking increases and decreases substantially in normal speech from instant to instant, even within short words, in a manner which varies from occasion to occasion. This results in the need to stretch or squeeze the time axis locally, which explains the name "time-warping". In addition, deletion and insertion may occur, because a speaker may slur "probably" to "prob'ly" or even "pro'lly", and expand the dictionary pronunciation "offen" into the spelling pronunciation "often" by inserting "t".

1e. Applications to gas chromatography. In gas chromatography (GC), the components of a gaseous mixture are separated from one another when the mixture is swept by a continuous stream of nonreactive carrier gas through a long densely packed column of special material, from which the components emerge at different times over a period of minutes or hours. The mixture itself enters the column all at once, and progress of different components through the column varies according to the duration and frequency of attachment of the different gas molecules to the packing material. In pyrolysis GC, the gaseous mixture is formed by rapidly heating input material, possibly microorganisms or cells, to a high temperature in the stream of carrier gas.

The amount of material (other than carrier gas) emerging from the column may be measured by a flame ionization detector and plotted as a function of time, to yield a chromatogram. Alternatively, a mass spectrometer may be used to provide detailed information about the emerging material every few seconds. Reiner et al. (1979, 1978, 1969) have applied these methods to identification of microorganisms and tissue samples.

However, comparison of chromatograms is sometimes hindered by nonlinear distortions of the time axis, due to nonuniform packing of the column material and nonuniform flow of the carrier gas. In some situations it is feasible to compensate for this distortion by sliding one chromatogram relative to the other

This content downloaded from 101.78.168.41 on Sat, 25 Nov 2017 15:20:11 UTC
All use subject to http://about.jstor.org/terms

as different portions are compared, but "such successes not withstanding, visual comparison of profiles over a light box proved to be a tedious experience" (Reiner et al., 1979, p. 491). Furthermore, if the multidimensional information provided by the mass spectrometer is to receive effective use, this simple method is not possible.

Time-warping offers an approach to compensating for time distortion which is automatic and can handle as much of the mass spectrogram as the experimenter wishes. Reiner et al. (1979) includes preliminary results which illustrate this use of time-warping.

1f. Applications to bird song. Bird song has been the subject of hundreds of papers in recent years. One reason for interest is that in some bird species, song is an important means of communication which is learned by the young from their elders and which has dialect-like variation from place to place. Such a phenomenon, which shares some important characteristics with human language, is rare among nonprimates. As explained in Section 8, the concept of "distance" between songs permits the useful application of quantitative methods in this field, and avoids serious difficulties which face a more straightforward approach to quantitative analysis. Bradley and Bradley [*] illustrates the use of such distances, and compares simple linear time-warping with dynamic time-warping, to the advantage of the latter.

1g. Other discrete-time applications: geological strata, tree-rings, varves, and text collation. In geology, stratigraphic sequences can be obtained from drill-holes and outcroppings. Each stratum may simply be classified as to type, or the classification may be supplemented by additional information about thickness, geochemical or mineral assay, fossils, etc. "Correlation" of different sequences is frequently necessary, i.e., it is necessary to match which strata in one such sequence correspond geologically to which strata in another. In the simplest case, this can easily be accomplished by a method which geologists call "cross-association". However, the sequences sometimes contain gaps or repetitive parts, due to local nondeposition, eroded strata, faulting, and folding, and the problem becomes more difficult. Smith and Waterman (1980) describe how to apply sequence comparison techniques to achieve correlation of sequences, and present two brief applications.

In principle, a great variety of other applications may exist similar to the one above, since strata are laid down by many natural processes. Two possible applications are dendrochronology and varve chronology. Dendrochronology refers to the science of dating based on tree rings (see, e.g., Ferguson, 1970). It requires the comparison of sequences obtained from different logs, most notably from logs of the bristle-cone pine. Dates possibly accurate to within one year have been obtained by this means further back than 5000 B.C. Varves are annual layers of sediment, generally clay, in which it is possible to count the years, typically due to variation between the summer and winter sediment (e.g., see Hörnsten, 1970, Fromm, 1970, and Tauber, 1970). Varve chronology requires comparison of sequences from different cores or different locations. It has been extended further back than 10,000 B.C.

Collation of different versions of the same text offers another possible application, as discussed in Cannon (1976, 1973).

This content downloaded from 101.78.168.41 on Sat, 25 Nov 2017 15:20:11 UTC
All use subject to http://about.jstor.org/terms

1h. Other continuous-time applications: handwriting and evoked potential waves. Time-warping can be used in computer processing handwritten material such as signatures, line drawings, and other handwritten material. Typically, the pen-point is tracked during the act of writing or drawing, with pressure of pen on paper or height of pen as a third coordinate, so that the record consists of the pen point trajectory as a function of time. Time-warping is important to compensate for nonlinear variations in writing speed. Handwriting recognition is addressed by Fujimoto et al. (1976) and Burr (1980), verification of signatures by Yasuhara and Oka (1977) and other related topics by Burr (1979, 1981).

There are many other potential applications of the same general kind, since many processes slow down and speed up irregularly. For example, brain evoked potentials ("brain waves" in response to a stimulus) are known to vary somewhat in timing from one occasion to another, and might constitute an application of time-warping.

1i. Visual space-warping for human depth perception. While not an application in the usual sense, the automatic, nonconscious process by which the human visual system fuses images from the two eyes during binocular depth perception is a form of sequence comparison. For simplicity, consider only black and white images that lie along a narrow strip in the horizontal direction (i.e., parallel to the line connecting the eyes). Each image is then a pair of functions $f_i(x_i)$, where the value of f indicates darkness, i indicates left or right eye, and x_i indicates horizontal position. It is known that the process of fusion is achieved by matching up values of x_1 with those of x_2 so that corresponding values of f_1 and f_2 agree. Suppose, as is usually true, that the two images differ only because the eyes view a three-dimensional scene from slightly different positions. Consider that which is visible as a single surface of varying darkness whose depth (i.e., distance from the eyes) also varies.

If depth varies only gradually, then fusion requires only continuous "space-warping" by means of compression and expansion. If depth has abrupt variations, as at the edge of a foreground object, then deletion and insertion are also required in order to deal with background regions blocked from one eye or the other by the foreground. Julesz (1971, 1978) has demonstrated that most people can experience a compelling sensation of depth when viewing "random-dot stereograms" that contain no clues to depth other than those depending on fusion, and has used such stereograms to investigate many aspects of vision.

2. How Sequences Differ

Different exemplars of what is nominally the same sequence are often subject to differences, such as

substitutions (also called replacements),
deletions and insertions (also called indels),
compression and expansions,
transpositions (also called swaps).

For example, on some occasions molecules of the same protein differ slightly among organisms of the same species: such differences are sometimes part of

This content downloaded from 101.78.168.41 on Sat, 25 Nov 2017 15:20:11 UTC
All use subject to http://about.jstor.org/terms

evolutionary change. If the same message is transmitted repeatedly on the same communication channel, it will be received differently on different occasions due to noise on the channel. The same word or computer instruction typed on different occasions may differ occasionally, due to human error. In speech, different utterances of the same word differ far more than most people realize, due not only to regional accent, personal idiosyncrasy, function in the sentence, and stress placed on the word, but also speed of speech, mood of the speaker and the inevitable unexplained factors which are lumped together under the heading of "random variation".

Differences should not be surprising. Rather in many examples, it is remarkable that the differences are so small. That many individual exemplars of the "same" macromolecule or repeated utterances of the "same" word are sufficiently similar to be recognized as the same, that is remarkable. Such close similarity can only be maintained by complex and subtle mechanisms.

Nevertheless, differences are universal. The mechanisms which underlie these differences vary greatly from one situation to another, but there is a common thread. In all our application areas, nominally same sequences have only *local* differences. For example, the beginning of one sequence corresponds roughly to the beginning of the other sequence, despite the differences, and does not correspond to the end of the other sequence.

If we compare two sequences, the most obvious type of difference between them is the substitution of one element (or term, or component, or unit) for another at the same position in the sequence. Such differences are called *substitutions* or *replacements* and are very common in most of the application areas. There are other important types of differences, however. These include *deletion* of elements and *insertion* of elements, and also *compression* (of two or more elements into one element) and *expansion* (of one element into two or more elements). *Dealing with differences between sequences due to deletion-insertion, compression-expansion, and substitution is the central theme of sequence comparison.* Both deletion-insertion and compression-expansion can conceal the precise correspondence between elements of nominally the same sequences.

Compression-expansion arises primarily in connection with sequences obtained by time-sampling from continuous functions of time, as in the processing of human speech. Compression-expansion is closely similar to deletion-expansion --so similar, in fact, that the differences have often been overlooked, and are clearly described for the first time in Kruskal and Liberman [*]. We will describe the differences briefly at the end of Section 4, when we have the machinery to discuss them. As a result of the similarity, it is possible to extend much of the material dealing with either of these topics to the other. In fact, a synthesis incorporating both types of difference at the same time is not difficult and is potentially useful. For historical reasons, however, much of the following material is phrased only in terms of deletion-insertion or only in terms of compression-expansion, rather than in terms of both.

Another type of local change, which is treated in Lowrance and Wagner (1975) and Wagner [*], is a *transposition* or *swap*, which means the interchange of two adjacent elements of a sequence. These are particularly relevant to human keyboard errors, but are also observed in linguistics under the name "metathesis".

This content downloaded from 101.78.168.41 on Sat, 25 Nov 2017 15:20:11 UTC
All use subject to http://about.jstor.org/terms

3. Analysis Of Differences: Trace, Alignment, And Listing

A basic approach of this field is to analyze the total difference between two sequences into a collection of individual elementary differences. The following discussion is limited to the case where the elementary differences are substitutions, deletions, and insertions. It will be helpful to have a single name to cover both insertions and deletions, so we coin the word *indel* from *in*sertion-*del*etion for this purpose. It is sometimes convenient to treat the elementary differences as *elementary operations*, and to think of the operations as actively changing a *source* sequence into a *target* sequence, step by step. Indeed, the very names "substitution", "insertion", etc. connote this active operational viewpoint.

At least three different modes of presentation for such analyses are in use: a *trace*, an *alignment*, and a *listing*. These are illustrated in Figure 1 and explained below. It is not generally realized that these are actually different modes of *analysis* as well as different modes of presentation, based on distinctions which will be made clear below. The distinctions are not usually exploited, however, which may be why they are not more commonly noticed.

In addition, for any given mode, many analyses of the same total difference are available: this is illustrated in Figure 2 using traces. The multiplicity of alternative analyses is one of the central difficulties of this field. A parsimony principle will be introduced in the next section to select the "shortest" or "least cost" analysis. Part of the multiplicity is an elaboration of the simple fact that

substitution of a by b

can also be analyzed as a deletion-insertion pair,

deletion of a and insertion of b.

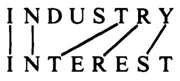
Which analysis seems more plausible depends on the context.

In parts of at least two major application areas, macromolecules and computer science, listings correspond directly to the natural mechanisms by which sequences are believed to change, but alignments (for macromolecules) and traces (for computer science) are typically used for analysis and presentation. One of the reasons for this is that the different kinds of analysis yield essentially the same results in many situations, but computation based on alignments or traces is far faster than computation based on listings. Thus listings are primarily of theoretical interest, while alignments and traces are used in practical work.

A *trace* from **a** to **b**, as illustrated in Figure 1, consists of the *source* sequence **a** above and the *target* sequence **b** below, usually with lines from some elements in the source to some elements in the target. An element can have no more than one line, and the lines must not cross each other (specifically the source elements with lines must correspond in order to the target elements with lines). The lines provide a correspondence, often partial, between source and target, and thereby supply a possible correspondence to fill the gap mentioned at the beginning of the paper. If the elements connected by a line are the same, we refer to the pair of elements as an *identity* or a *continuation* (because the target element "continues" the source element); if they are different, the pair constitutes a *substitution*; in either case we call the pair a *match*. A source element having no line shows a

This content downloaded from 101.78.168.41 on Sat, 25 Nov 2017 15:20:11 UTC
All use subject to http://about.jstor.org/terms

Trace



Alignment or *Matching*

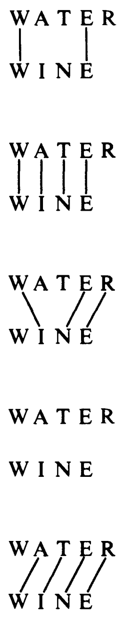
$$\begin{bmatrix} I & N & D & U & S & T & \varnothing & R & \varnothing & Y & \varnothing \\ I & N & \varnothing & \varnothing & \varnothing & T & E & R & E & S & T \end{bmatrix}$$

Listing

INDUSTRY	
	Delete D
INUSTRY	
	Delete U
INSTRY	
	Substitute Y by S
INSTRS	
	Insert E
INSTERS	
	Insert E
INSTERES	
	Delete S
INTERES	
	Insert T
INTEREST	

THREE MODES FOR ANALYZING DIFFERENCES BETWEEN SEQUENCES.

FIG. 1.



DIFFERENT ANALYSES OF THE SAME PAIR OF SEQUENCES

FIG. 2.

This content downloaded from 101.78.168.41 on Sat, 25 Nov 2017 15:20:11 UTC
All use subject to http://about.jstor.org/terms

deletion; a target element having no line shows an *insertion*; either of these is an *indel*. See Figure 3 for a summary of this terminology. In the trace of Figure 1,

I, N, T, and R of the source are continued,
Y of the source is substituted by S of the target,
D, U, and S of the source are deleted,
E, E, and T of the target are inserted.

An *alignment* (or *matching*) between **a** and **b**, as illustrated in Figure 1, consists of a matrix of two rows. The upper row consists of the *source* **a**, possibly interspersed with *null characters* (or *nulls* for short), which are represented here by ϕ. (Some authors use λ or $-$ or a blank for the same purpose.) The lower row consists of the *target* sequence **b**, possibly interspersed with null characters. The column $\left[\begin{smallmatrix}\phi\\ \phi\end{smallmatrix}\right]$ of null characters is not permitted. A column $\left[\begin{smallmatrix}x\\ \phi\end{smallmatrix}\right]$ having ϕ below indicates *deletion;* a column $\left[\begin{smallmatrix}\phi\\ y\end{smallmatrix}\right]$ having ϕ above indicates *insertion;* either of these is an *indel.* A column $\left[\begin{smallmatrix}x\\ y\end{smallmatrix}\right]$ without ϕ is called a *match;* if $x \neq y$ it is a *substitution;* if $x = y$ it is called a *continuation.* See Figure 3 for a summary of this terminology. The matches supply a correspondence, often only partial, between source and target, thereby supplying a possible correspondence to fill the gap mentioned at the beginning of the paper. The alignment in Figure 1 corresponds to the trace in the figure, and has the corresponding continuations, substitutions, etc.

As a mode of analysis, alignments are richer than traces in the sense that an alignment makes some order distinctions between adjacent indels which a trace does not, as Figure 4 illustrates. There is a simple construction for transforming alignments into traces by abandoning the order restrictions (which transforms all the alignments of Figure 4 into the trace there). The construction has two steps. First, copy the source and the target from the alignment, one above the other, i.e., copy the top and bottom rows of the alignment but omit the nulls. Then for each match in the alignment, draw a line in the trace to connect the top and bottom elements it provides. The resulting trace has the same source, target, continuations, substitutions, deletions, and insertions as the alignment. (In mathematical terminology, the mapping from alignments to traces is many-to-one and onto.)

In a great many situations the order of adjacent insertions and deletions is unimportant. If so, traces can be used for analysis, or alignments can be used in a way which ignores the irrelevant aspects of the ordering. On the other hand, there are some situations in which the order of adjacent indels is important, and traces are not adequate. For example, this applies to the constraint that no more than F consecutive deletions and no more than G consecutive insertions are permitted. It also applies to the constraint that no more than K strings of indels are permitted, where each string consists either of consecutive deletions or of consecutive insertions (with no limit on the length of each string). These and other constraints are treated in Kruskal and Sankoff [*].

As modes of presentation (rather than modes of analysis), traces and alignments each have their own advantages. Diagrams which are basically alignments, though often improved by inclusion of auxiliary information, have been favored in macromolecular applications. (For such practical use, a more inconspicuous null character than ϕ should be used, e.g., — or a blank). Traces have been favored in some computer science applications.

A *listing* from **a** to **b**, as illustrated in Figure 1, consists of an alternating series of sequences and elementary operations, starting with the *source* sequence **a**

This content downloaded from 101.78.168.41 on Sat, 25 Nov 2017 15:20:11 UTC
All use subject to http://about.jstor.org/terms

TYPE OF OPERATION

		Match		Indel	
		No action	Elementary operations		
	DESCRIPTION	Identity or Continuation	Substitution or Replacement	Deletion	Insertion
MODE OF ANALYSIS	Trace	Line joining identical elements	Line joining different elements	Element in source with no line	Element in target with no line
	Alignment or Matching	Column with same element above and below	Column with different elements above and below	Column with ∅ below	Column with ∅ above
	Listing	Equal source and target elements which can be traced to each other	Unequal source and target elements which can be traced to each other	Source element which cannot be traced to target	Target element which cannot be traced to source

DESCRIPTION OF OPERATIONS

FIG. 3

This content downloaded from 101.78.168.41 on Sat, 25 Nov 2017 15:20:11 UTC
All use subject to http://about.jstor.org/terms

Trace

$$\begin{tikzcd}[row sep=small, column sep=tiny] g \arrow[d, dash] & a & b & h \arrow[d, dash] \\ g & c & d & h \end{tikzcd}$$

Alignments

$$\begin{bmatrix} g & a & b & - & - & h \\ g & - & - & c & d & h \end{bmatrix}$$

$$\begin{bmatrix} g & a & - & b & - & h \\ g & - & c & - & d & h \end{bmatrix}$$

$$\begin{bmatrix} g & a & - & - & b & h \\ g & - & c & d & - & h \end{bmatrix}$$

$$\begin{bmatrix} g & - & a & b & - & h \\ g & c & - & - & d & h \end{bmatrix}$$

$$\begin{bmatrix} g & - & a & - & b & h \\ g & c & - & d & - & h \end{bmatrix}$$

$$\begin{bmatrix} g & - & - & a & b & h \\ g & c & d & - & - & h \end{bmatrix}$$

SIX ALIGNMENTS WHICH CORRESPOND TO THE SAME TRACE

FIG. 4.

and ending with the *target* sequence **b**, which satisfies the following consistency requirement:

Listing consistency property. Two adjacent sequences in the listing must differ only as provided by the intervening elementary operation.

Thus a listing is in effect an algorithm which describes how to change the source into the target. Listings are primarily of theoretical interest, rather than for direct use in applications, and are a much richer mode of analysis than alignments or traces.

For any two successive sequences in a listing, consider their elements, but exclude the element if any which was inserted or deleted by the intervening elementary operation. Then there is an obvious correspondence between the elements in one sequence and those in the other. By using such correspondences, it is possible to trace forwards and backwards through the listing. Any source element which cannot be traced all the way to the target sequence is called an *overall deletion*; any target element which cannot be traced all the way to the source is called an *overall insertion*; both of course are *overall indels*. (The modifier

This content downloaded from 101.78.168.41 on Sat, 25 Nov 2017 15:20:11 UTC
All use subject to http://about.jstor.org/terms

"overall" is used to indicate a distinction, which did not exist for alignments and traces, between elementary operations which are components of the listing and those which are the net result of it.) For any element in the source sequence which can be traced all the way to the target sequence, consider the pair consisting of the source element and its corresponding target element. This pair is called an *overall match*; if the two elements are equal, it is called an *overall identity* or an *overall continuation*; if they are unequal, it is called an *overall substitution*. See Figure 3 for a summary of this terminology. The overall matches supply a correspondence, often only partial, between source and target, and thereby supply a possible correspondence to fill the gap mentioned at the beginning of the paper. The listing in Figure 1 corresponds to the alignment and the trace shown there, and has the corresponding continuations, substitutions, etc.

The operations found formed by tracing forward and backward from any single element involve at most one insertion, an unlimited number of substitutions, and at most one deletion. The order of these operations within the listing is important of course: none of them may precede the insertion or follow the deletion, and substitution $A \to B$ can immediately precede but not immediately follow the substitution $B \to C$. On the other hand, if the order of operations which act on different positions is changed, then it is a simple matter to adjust the intervening sequences so that the listing consistency property is preserved.

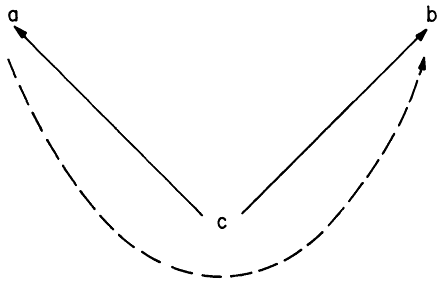
As a practical mode of presentation, listings are awkward and have been little used. As a mode of analysis, listings have theoretical importance because it is possible to generalize them much more broadly than alignments and traces, and because they correspond to plausible underlying mechanisms in several major applications. They are richer than alignments because listings permit many successive changes to be made in a single position and alignments permit only one. In addition, listings make distinctions based on the order in which changes are made, and alignments do not. There is a simple natural construction for transforming listings into alignments by abandoning these distinctions. The first step is to copy the source and target from the listing, spacing them so that each overall match is lined up vertically and forms a column. Then a null character is placed in the blank space below each overall deletion and in the blank space above each overall insertion. The resulting two rows are an alignment which has the same source, target, continuations, substitutions, etc. as the listing. (In mathematical terms, the mapping from listings to alignments is many-to-one and onto.) Listing distance underlies the approach to transpositions used in Wagner [*], though in practice Wagner works with a simpler equivalent distance based on generalized traces.

In practice part of the richness of listings is seldom available, namely, the possibility of making several successive changes in one position. Under assumptions which are usually applicable, listings with more than one change in a position will not be selected for use due to the parsimony principle to be introduced in the next section.

A listing resembles a description of the individual changes which might have taken place during some tangible process of change, such as evolution from one sequence to another, or processing by the nervous system of a planned sequence of motor activities, through several stages from formation to execution. Although this resemblance is legitimately exploited in some applications, it is important that great caution be used in any attempt to interpret listings in tangible fashion, for

This content downloaded from 101.78.168.41 on Sat, 25 Nov 2017 15:20:11 UTC
All use subject to http://about.jstor.org/terms

several reasons. (1) Listings may be formed and used without regard for whether the situation makes it sensible to think of changes as taking place in a series of listing-like steps. For example, in the speech research area it is not easy to make such an interpretation, yet distances (see the next section) based on listings or other analyses are indubitably very useful. (2) During an historical process such as evolution, more than one change in a position is a very real possibility, but (as explained above) the parsimony principle will usually prevent the listing from having this. (3) The operations occur in some definite order in the tangible process of change, and they occur in a definite order in the listing, but there seldom is evidence which permits the listing order to reconstruct the tangible process order. (4) In the usual evolutionary situation, sequence **a** is not the ancestor of sequence **b**, but instead both **a** and **b** are descendants of some unknown common ancestor **c** (see Figure 5). If a tangible interpretation is given to the steps from **a** to **b** (dotted path in figure), some of them presumably are the steps from **a** to **c** *in reverse* (insertions become deletions, and deletions insertions, e.g.) while others are the steps from **c** to **b** (in normal manner).



a,b ARE OBSERVED SEQUENCES
c IS UNOBSERVED ANCESTOR

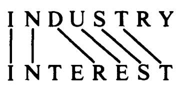
USUAL EVOLUTIONARY SITUATION

FIG. 5

4. Levenshtein And Other Distances

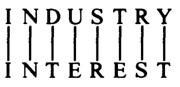
Levenshtein (1965a) introduced two closely related concepts of distance $d(\mathbf{a},\mathbf{b})$ from one sequence **a** to another sequence **b**. One of these is the smallest number of substitutions and indels required to change **a** into **b**. The other is the smallest number of indels (no substitutions permitted) required to change **a** into **b**. For example, it is possible to change **a** = INDUSTRY into **b** = INTEREST using 7 substitutions and indels, as shown in Figure 1. However, it is also possible to change **a** to **b** using only 6. Two of the many ways in which this can be done are shown in Figure 6, as listings, alignments and traces. It can be shown that 6 is the minimum, so one kind of Levenshtein distance from INDUSTRY to INTEREST is 6. It turns out that the minimum number of indels to make the same change is 8, so the other kind of Levenshtein distance is 8.

This content downloaded from 101.78.168.41 on Sat, 25 Nov 2017 15:20:11 UTC
All use subject to http://about.jstor.org/terms



$$\begin{bmatrix} \text{I N - - D U S T R Y} \\ \text{I N T E R E S T - -} \end{bmatrix}$$

INDUSTRY	
	Delete Y
INDUSTR	
	Delete R
INDUST	
	Substitute D by R
INRUST	
	Substitute U by E
INREST	
	Insert T
INTREST	
	Insert E
INTEREST	



$$\begin{bmatrix} \text{I N D U S T R Y} \\ \text{I N T E R E S T} \end{bmatrix}$$

INDUSTRY	
	Substitute D by T
INTUSTRY	
	Substitute U by E
INTESTRY	
	Substitute S by R
INTERTRY	
	Substitute T by E
INTERERY	
	Substitute R by S
INTERESY	
	Substitute Y by T
INTEREST	

SOME SHORTEST ANALYSES

FIG. 6.

These concepts are only two of a wide class of meanings which are given to the word "distance" in sequence comparison. In preparation for discussing these many meanings, we rephrase the first preceding definition in terms of four parts.

(i) Let the elementary operations be substitutions and indels.

(ii) Consider all listings from **a** to **b** based on the elementary operations.

(iii) Let the length of each listing be the number of elementary operations it contains.

(iv) Then the distance is the minimum length of any listing.

The most obvious way to achieve the other definition in this framework is to reduce the elementary operations to indels only, and this is very much within the larger spirit of the field. Alternatively, we can leave the elementary operations unchanged, but redefine the length of a listing to be

$$\text{number of indels} + w_s \text{ (number of substitutions)}, \quad \text{with } w_s \geqq 2.$$

(If $w_s > 2$, then it is always shorter for a listing to use a deletion-insertion pair in place of a substitution, and if $w_s = 2$ it is as short.)

Many meanings of distance in sequence comparison can be described using four parts like those above.

(1) *Elementary operations.* Each definition is based on some set of *elementary differences.* In the following discussion, the elementary differences are

This content downloaded from 101.78.168.41 on Sat, 25 Nov 2017 15:20:11 UTC
All use subject to http://about.jstor.org/terms

substitutions and indels. Elsewhere, other elementary differences are added to the list, such as transposition, or compression and expansion.

(2) *Acceptable analyses.* Each definition is based on some class of *acceptable analyses* of the difference between **a** and **b**, where the acceptable analyses are based on the elementary differences. In the following discussion, acceptable analyses may be all listings (from **a** to **b**), or all alignments, or all traces. Other possibilities are used elsewhere, such as a constrained set of alignments (e.g., those in which no more than 2 consecutive deletions and no more than 2 consecutive insertions occur), or all time-warpings.

(3) *Length.* Each definition is based on some function for calculating the *length* of an analysis. The function may incorporate a system of weights or parameters. The "simple length" function (described below) is used here and in most applications based on substitutions and indels. Other length functions, including those used with compression-expansion, are discussed below.

(4) *Parsimony principle.* In general, the distance $d(\mathbf{a},\mathbf{b})$ is defined to be the minimum length of any acceptable analysis of the difference between **a** and **b**.

Algorithms for performing the minimization referred to in (4) are a central theme in sequence comparison. As different meanings of length are considered, different algorithms may be needed, and algorithms to do the minimization are almost always an important consideration. The computing time required may be important, and often computer storage requirements are too. Algorithms are discussed later in the paper.

Suppose the elementary differences consist of substitutions and indels. One widely used length function, which we call *simple length*, is based on weights for the elementary differences. Specifically, we assume that each possible substitution and indel has a given weight $\geqq 0$. The simple length of an analysis is the sum of the weights of the elementary operations it contains (in case of a listing, this refers to the operations in the individual steps, not the overall operations). If all the weights are 1, then the simple length function simply counts the number of elementary operations, and we obtain one of Levenshtein's original distances. If the weight of every indel is 1 and the weight of every substitution is $w_s \geqq 2$, we obtain the other. We use the following notation for the weights:

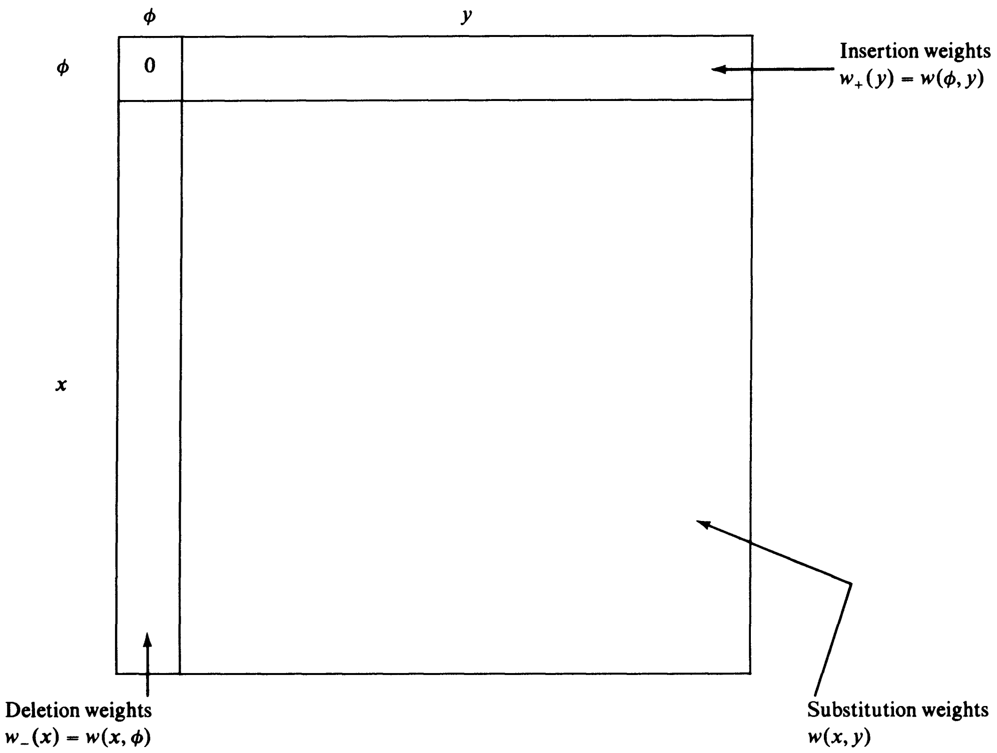
Substitution $x \longrightarrow y$	$w(x,y)$
Deletion of x	$w(x,\phi)$ or $w_{\text{del}}(x)$ or $w_{-}(x)$
Insertion of y	$w(\phi,y)$ or $w_{\text{ins}}(y)$ or $w_{+}(y)$

When the null character ϕ is permitted as an argument of w, we refer to *extended* w and use the convention $w(\phi,\phi) = 0$. If the alphabet of possible sequence elements is finite, the weights may be arranged into a matrix (see Figure 7) where x corresponds to the row and y to the column.

Any given extended w (and any given class of acceptable analyses) leads to a distance function d. For any elements x and y, it is possible to compare

$$\begin{array}{lll} w(x,y) & \text{with} & d(x,y), \\ w(x,\phi) & \text{with} & d(x,\phi) \\ w(\phi,y) & \text{with} & d(\phi,y), \end{array}$$

This content downloaded from 101.78.168.41 on Sat, 25 Nov 2017 15:20:11 UTC
All use subject to http://about.jstor.org/terms



EXTENDED MATRIX OF WEIGHTS

FIG. 7.

since any element is also a length 1 sequence and the null character is the length 0 sequence. It may turn out that w and d are equal in all such comparisons: if so, we say that d *agrees with* w. When this happens, we can regard w as being part of d, namely, w is d restricted to sequences of length 1 or 0. The process of deriving d from w can be regarded as a process of *extending* the definition of distance from the restricted case of sequences of length $\leqq 1$ to all sequences. We shall refer to this approach as *extension of d from short sequences,* while the approach we have used will be called *definition of d from weights.* When the extension approach is used, the same letter (often d) is used for both w and d, and agreement of d with w becomes a form of self-consistency for d. In the extension approach, it is necessary to make assumptions which insure this self-consistency. Further discussion may be found in Kruskal and Sankoff [*].

We shall use *Levenshtein distance* to cover any distance based on the simple length function. (Existing usage is not entirely clear about the meaning of this phrase, but this definition seems to match usage pretty well.) If the simple length function is used with all listings, or with all alignments, or with all traces as the acceptable analyses, in principle we obtain three different distance functions. However, alignments and traces yield the same distance function in this context,

This content downloaded from 101.78.168.41 on Sat, 25 Nov 2017 15:20:11 UTC
All use subject to http://about.jstor.org/terms

so we only refer to *simple listing distance* and *simple alignment distance*. Even these often turn out to be the same (see Kruskal and Sankoff [*]).

In the mathematical literature, the word "distance" is ordinarily used to indicate a function d which satisfies the *metric axioms*: (i) *Nonnegative property*, $d(\mathbf{a},\mathbf{b}) \geqq 0$ for all **a** and **b**; (ii) *Zero property*, $d(\mathbf{a},\mathbf{b}) = 0$ if and only if $\mathbf{a} = \mathbf{b}$; (iii) *Symmetry*, $d(\mathbf{a},\mathbf{b}) = d(\mathbf{b},\mathbf{a})$ for all **a** and **b**; (iv) *Triangle inequality*, $d(\mathbf{a},\mathbf{b}) + d(\mathbf{b},\mathbf{c}) \geqq d(\mathbf{a},\mathbf{c})$ for all **a**, **b**, **c**. Although we do not restrict the word "distance" in this way, we are interested in these important properties. It is common to make suitable assumptions on w in order to achieve the metric axioms for the distance d which is derived from w. Of course, the assumptions needed for this purpose depend on which type of sequence distance is being used. We remark that the triangle inequality always holds for simple listing distance d, whether or not it holds for w, but the same is not true for simple alignment distance.

Other length functions. The simple length function described above covers most applications based on indels. There are, however, at least three other types of length functions in use. The most important one is used in connection with time-warping, where compression-expansion plays a central role. It is described last here, because it requires the most space. Another directly generalizes the simple length function, to meet an applied need. Finally, one very unusual length function, used for an application which does not fit the framework above, may be found in Sankoff, Kruskal, Mainville, and Cedergren [*].

One reason for using distance in macromolecular applications, as described below in Section 9, is that it bears an approximate inverse relationship to the likelihood of one sequence changing into another in some circumstances. In some applications it may be possible for whole strings to be deleted or inserted, with the likelihood decreasing only slowly as the number of elements in the string increases (so that a single deletion of two adjacent elements is far more likely than two deletions of separated elements). To achieve a reasonable correspondence between distance and likelihood in this situation, it is appropriate to use a length function in which the dominant term is a weight for each consecutive string of deletions or insertions, and in addition much smaller weights used (as in simple length) for each single element deletion or insertion. Further information about this length function may be found in Kruskal and Sankoff [*].

In applications using compression-expansion, where sequence comparison is generally referred to as time-warping, the concept of simple length is of course not meaningful. An adequate discussion of the length functions used with time-warping would require too much space to include here, but we shall briefly illustrate the ideas.

We noted above in Section 2 that compression-expansion is extremely similar to deletion-insertion. It turns out that the most important differences lie in the nature of the length functions which are appropriate to use. In particular, let us consider expansion of one element in **a** (say, a_6) into two adjacent elements of **b** (say, b_6,b_7). It is possible to achieve the same change with substitutions and indels, e.g., insertion of b_7 together with substitution of a_6 by b_6. (The substitution may be omitted if it happens that $a_6=b_6$.) The most natural cost to assign to the combination of insertion and substitution is the simple length involved, namely,

$$w_{\text{ins}}(b_7) + w(a_6,b_6) .$$

This content downloaded from 101.78.168.41 on Sat, 25 Nov 2017 15:20:11 UTC
All use subject to http://about.jstor.org/terms

On the other hand, the most natural cost to assign to the expansion is

$$\frac{1}{2} w(a_6,b_6) + \frac{1}{2} w(a_6,b_7) + w_{1,2} ,$$

where the former terms are a penalty for how different a_6 is from the elements it expanded into, and the latter term is a penalty for distorting the time-axis by matching 1 element with 2 elements. An example of the use of length functions for time-warping may be found in Hunt, Lennig, and Mermelstein [*], and a systematic discussion of such length functions may be found in Kruskal and Liberman [*]. Extensive applications of the time-warping concept may be found in the speech processing literature: for references see Section 1d above.

5. Calculating Distance: The Basic Algorithm

Distance from **a** to **b** is defined as the minimum length of any acceptable analysis of the difference between **a** and **b**. Efficient methods for doing the optimization to find the distance and the corresponding optimum analysis are a central topic of sequence comparison. In some applications it is the distance and in others it is the analysis which is of primary interest. If there happen to be several different optimum analyses, we may wish to know them all.

One concept for finding the minimum, which can be described as a form of dynamic programming, has been discovered independently and described several times (Vintsyuk, 1968; Needleman and Wunsch, 1970; Velichko and Zagoruyko, 1970; Sakoe and Chiba, 1970, 1971; Sankoff, 1972; Reichert, Cohen, and Wong, 1973; Haton, 1973, 1974a, 1974b; Bridle, 1973; Wagner and Fisher, 1974; Hirschberg, 1975; and perhaps others. The Needleman and Wunsch program is cited with a 1967 date in Barker et al., 1969). While the algorithms described in these papers do differ somewhat, they can all be seen as flowing out of the same concept. Different algorithms remain useful today for different situations. We describe here the simplest, most basic algorithm of this kind. Many other related algorithms appear throughout the literature.

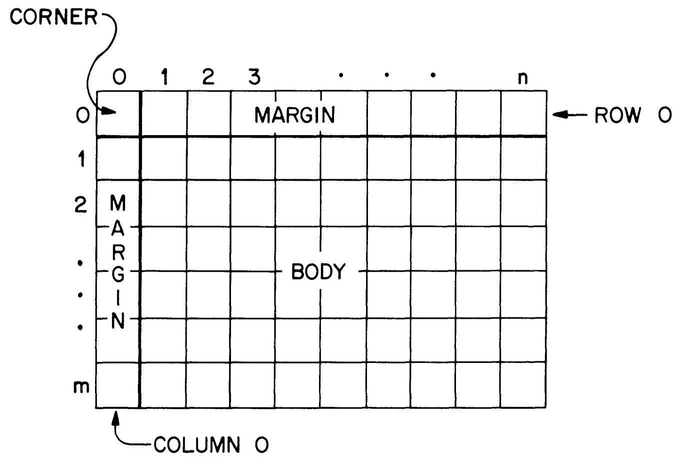
One essential common element in all the papers cited is reliance on distances akin to alignment or trace distance, as opposed to listing distance, since use of the latter would require far slower calculations. In many cases, listing distance is the conceptual starting point, though this fact may not be clearly brought out; in such cases, the mathematical equivalence of listing distance with alignment and trace distance is a vital foundation for practical sequence comparison. By way of contrast, consider Wagner [*], in which transpositions are introduced as elementary operations. Listing distance is the conceptual starting point, and in that context listing distance is *not* mathematically equivalent to trace distance as defined above, but to a complicated generalization of trace distance. This leads to great complications in the algorithm, which illustrates the importance of our comment.

The first stage of the algorithm finds simple alignment distance, and an optional second stage can be used to find an optimum alignment if desired. If the two sequences have length m and n, then the dominant portion of the computing time for this algorithm is proportional to mn, and the dominant portion of memory space is also. If $m = n$, this is n^2, so the algorithm requires computing time which is quadratic in the length of the sequences.

This content downloaded from 101.78.168.41 on Sat, 25 Nov 2017 15:20:11 UTC
All use subject to http://about.jstor.org/terms

For the sake of concreteness, the following description includes some details which are intended as an aid to understanding, not necessarily for actual use. Let the two sequences be **a** and **b**, with entries a_i and b_j, and having lengths m and n. Substitution weights are given by $w(x,y)$, deletion weights by $w(x,\phi)$, and insertion weights by $w(\phi,y)$. Let $\mathbf{a}^i$ and $\mathbf{b}^j$ indicate the initial segments $a_1 \cdots a_i$ and $b_1 \cdots b_j$ (including the case where the initial segment has length 0, namely $\mathbf{a}^0$ or $\mathbf{b}^0$, which is simply the empty sequence). Of course, $\mathbf{a}^m = \mathbf{a}$ and $\mathbf{b}^n = \mathbf{b}$.

In the first stage, we work forward recursively, and find distances $d(\mathbf{a}^i,\mathbf{b}^j)$ for successively larger i and j, finally reaching $d(\mathbf{a}^m,\mathbf{b}^n) = d(\mathbf{a},\mathbf{b})$, which is the distance desired. One procedure is to record these values in an array like that shown in Figure 8, where we start from the (0,0) cell in the upper left-hand corner, and move toward the (m,n) cell in the lower right. If the optional second stage is to be carried out, a necessary preparation during the first stage is to record certain pointers. The pointers are described in abstract form in the equation below, and portrayed visually as arrows in the array. The optional second stage, often referred to as "backtracking", consists of tracing along the pointers or arrows.



ARRAY USED TO PERFORM DYNAMIC PROGRAMMING ALGORITHM

FIG. 8

The recursion of the first stage starts from the obvious value $d(\mathbf{a}^0,\mathbf{b}^0) = 0$, which provides the entry for the (0,0) cell. The recurrence equation for a typical cell (i,j) is based on the values in three predecessor cells, $(i-1,j)$, $(i-1,j-1)$, and $(i,j-1)$. However, if $i = 0$ or $j = 0$, two predecessor values involving negative values of i or j are not used. No cell can be processed until its predecessors have been processed, but otherwise the order of calculation makes no difference.

This content downloaded from 101.78.168.41 on Sat, 25 Nov 2017 15:20:11 UTC
All use subject to http://about.jstor.org/terms

Now we give the recurrence equation which is used to calculate the (i,j) value, and the corresponding pointer equation which serves as preparation for stage two. The intuitive meaning will be described shortly:

$$d(\mathbf{a}^i,\mathbf{b}^j) = \min \begin{cases} d(\mathbf{a}^{i-1},\mathbf{b}^j) & + w(a_i,\phi), & (\text{"deletion of } a_i\text{"}), \\ d(\mathbf{a}^{i-1},\mathbf{b}^{j-1}) & + w(a_i,b_j), & (\text{"substitution of } a_i \text{ by } b_j\text{"}), \\ d(\mathbf{a}^i,\mathbf{b}^{j-1}) & + w(\phi,b_j), & (\text{"insertion" of } b_j\text{"}), \end{cases}$$

$$\text{pointer}(i,j) = \left\{ \begin{matrix} (i-1,j) & \text{or} \\ (i-1,j-1) & \text{or} \\ (i,j-1) & \end{matrix} \right\}, \quad \begin{matrix}\text{where choice depends on} \\ \text{the minimizing entry above.}\end{matrix}$$

The choice in the pointer equation simply records which line was selected as the minimum in the recurrence. In the minimization any line containing a negative value of i or j is simply ignored. The intuitive meaning of the recurrence formula is that the shortest alignment between $\mathbf{a}^i$ and $\mathbf{b}^j$ may be formed by choosing the shortest of the following three alternatives:

(i) use the shortest alignment between $\mathbf{a}^{i-1}$ and $\mathbf{b}^j$, and also delete a_i;

(ii) use the shortest alignment between $\mathbf{a}^{i-1}$ and $\mathbf{b}^{j-1}$, and also substitute a_i by b_j,

(iii) use the shortest alignment between $\mathbf{a}^i$ and $\mathbf{b}^{j-1}$, and also insert b_j.

One procedure for evaluating the recurrence equation is to start by recording some auxiliary values in the cell, as shown in Figures 9 and 10. For a body cell, these are the three w values, namely, $w(a_i,\phi)$, $w(a_i,b_j)$, and $w(\phi,b_j)$. For a margin cell, only one of these entries is used, of course. Next the d values from the preceding cell(s) are added to the corresponding w values, as indicated by double arrows in Figure 11, to form sums. For body cells, there are three preceding cells, three w's, and three sums, while for margin cells there is just one of each. Finally, the minimum of the sum is recorded as the new d value. The pointer equation records which one of the sums is minimum. Pictorially, the pointer is shown as a heavy single arrow in Figure 11: the other two possible heavy single arrows are shown in dotted form. If more than one of the sums provides the minimum value, this can be indicated in the pointer equation by using a set of two or three pairs on the right hand side, and pictorially by using two or three arrows.

To prove the validity of the recurrence equation, consider a shortest alignment between $\mathbf{a}^i$ and $\mathbf{b}^j$, whatever it may be. Since an alignment may not contain a column of null letters, the right-hand column must be either $\begin{bmatrix} a_i \\ \phi \end{bmatrix}$ or $\begin{bmatrix} \phi \\ b_j \end{bmatrix}$ or $\begin{bmatrix} a_i \\ b_j \end{bmatrix}$. (If $i=0$ or $j=0$, the entry a_i or b_j does not exist, and only one of these cases is possible.) Consider the first case. If we delete the last column from the alignment, the remaining alignment between $\mathbf{a}^{i-1}$ and $\mathbf{b}^j$ must be a shortest possible one, for if there were a shorter one it could be substituted, and this would yield a shorter alignment between $\mathbf{a}^i$ and $\mathbf{b}^j$. The simple length of the entire alignment is the length of the first portion, which must be $d(\mathbf{a}^{i-1},\mathbf{b}^j)$, plus the simple length of the last column, which is $w(a_i,\phi)$. This gives us the first line in the formula. Similar reasoning in the other two cases gives the other two lines

This content downloaded from 101.78.168.41 on Sat, 25 Nov 2017 15:20:11 UTC
All use subject to http://about.jstor.org/terms

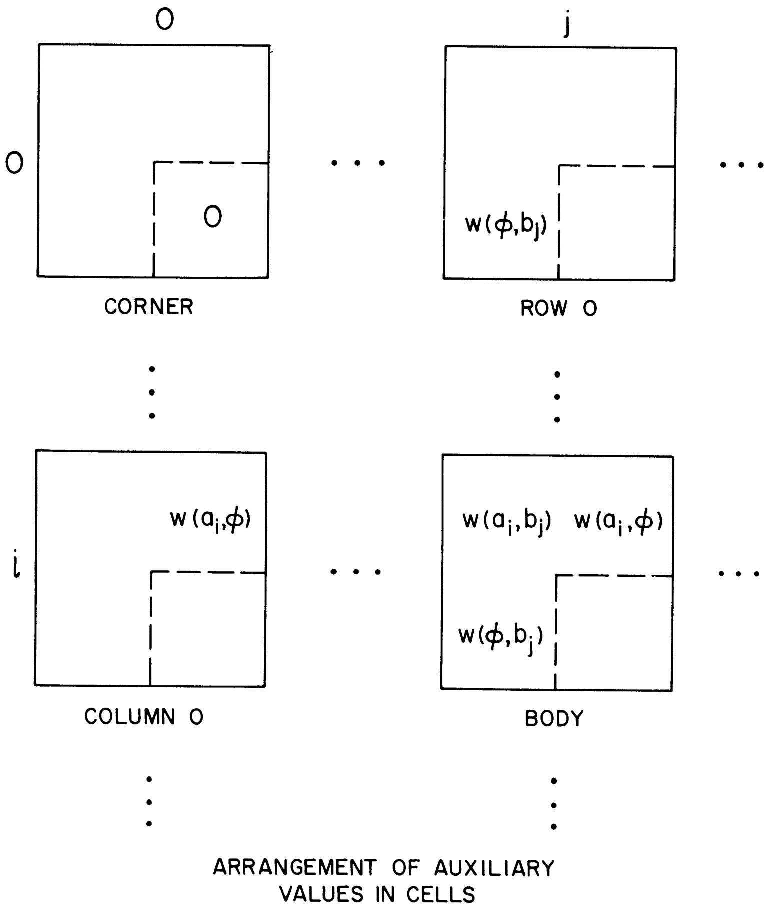


FIG. 9

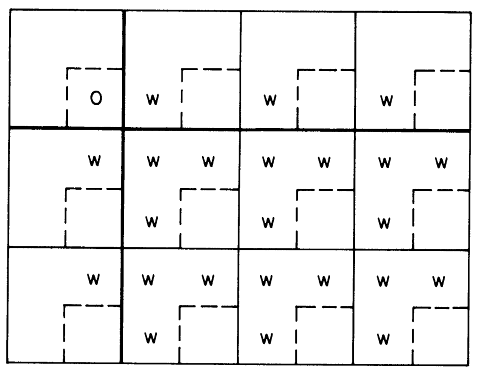
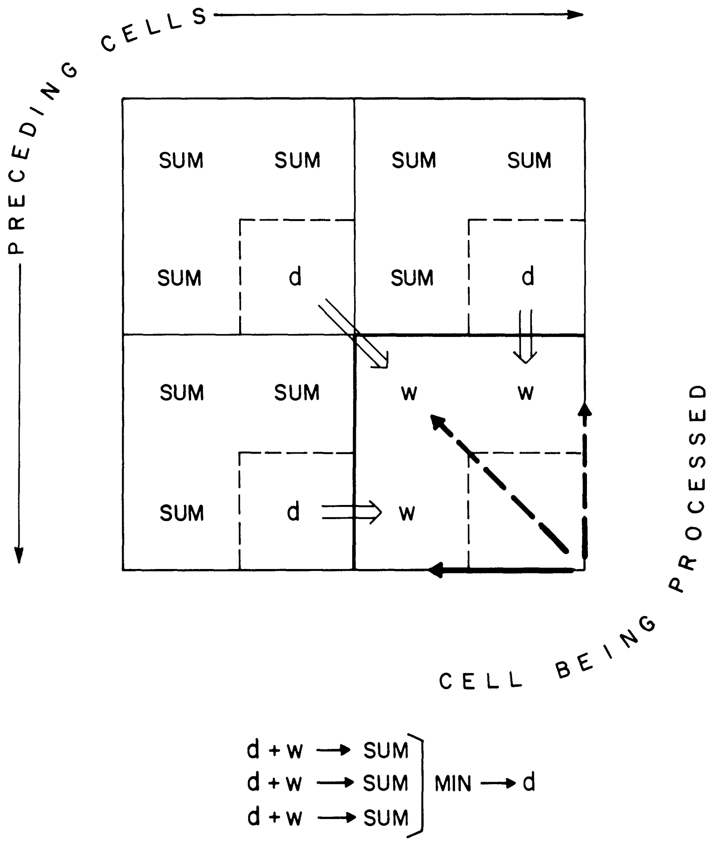


FIG. 10

This content downloaded from 101.78.168.41 on Sat, 25 Nov 2017 15:20:11 UTC
All use subject to http://about.jstor.org/terms



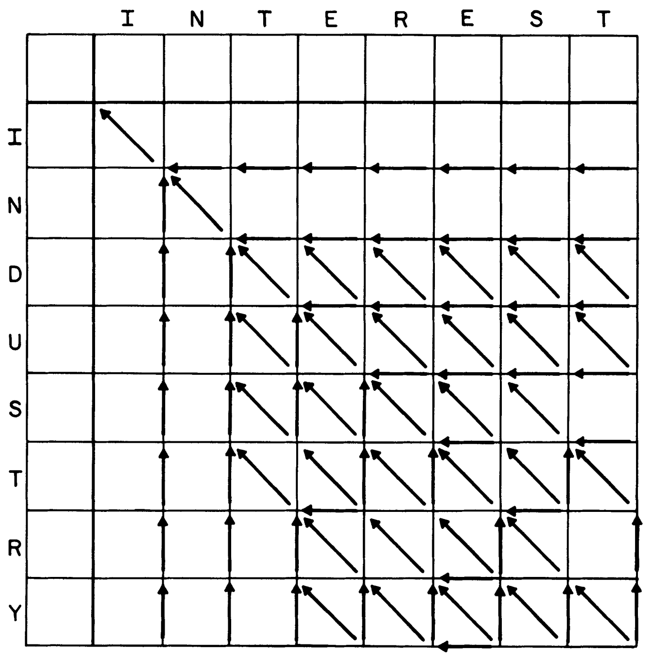
ONE STEP OF RECURRENCE

FIG. 11

in the formula. Thus $d(\mathbf{a}^i, \mathbf{b}^j)$ must be one of the three alternatives shown, and since d is defined to be the minimum possible alignment length, it must be the minimum of the three alternatives.

The second stage "backtracking" calculation, which finds the optimum alignment(s), is very simple. In pointer form, this consists of starting with the final cell (m,n), and using the pointer information repeatedly to obtain a path all the way to cell (0,0). Pictorially, as illustrated in Figure 12, this means following along the arrows. In either case, an alignment is read off from the path in reverse order in the obvious way, which we phrase in terms of the arrows: a diagonal arrow in cell (i,j) means a_i is substituted by b_j and $\begin{bmatrix} a_i \\ b_j \end{bmatrix}$ is placed in the alignment; a horizontal arrow in column j means insertion of b_j and $\begin{bmatrix} \phi \\ b_j \end{bmatrix}$ is placed in the alignment; and a vertical arrow in row i means deletion of a_i and $\begin{bmatrix} a_i \\ \phi \end{bmatrix}$ is placed in the alignment. If all optimum alignments are desired, all possible alternative paths should be traced out and used. Two of the alignments which can be formed from Figure 12 in this way are shown in Figure 6.

This content downloaded from 101.78.168.41 on Sat, 25 Nov 2017 15:20:11 UTC
All use subject to http://about.jstor.org/terms



ARRAY USED FOR TRACING BACKWARDS

FIG. 12

6. Computational Complexity Of Sequence Comparison

Computational complexity has been investigated for several problems in sequence comparison. (The computational complexity of a problem means the minimum possible computation time that is needed to solve a worst-case example of that problem.) Consider the problem of calculating the optimal alignment between two sequences of length n. Since the algorithm given in the preceding section requires time proportional to n^2, we know the computational complexity is not greater than proportional to n^2. Using fairly general assumptions, Wong and Chandra (1976) proved that the computational complexity is in fact proportional to n^2.

However, computational complexity is often very sensitive to the precise assumptions. Masek and Patterson [*] present an algorithm which requires time only proportional to $n^2/\log(n)$, based on two mild assumptions: the sequence elements come from a finite alphabet, and the weights (of substitutions and indels) are all rational. Asymptotically, their method is the fastest one currently available, but it has not been proved optimal. As a practical matter, Masek and Patterson point out that their method does not become faster than a simple method similar to that in the preceding section until $n \geqq 262{,}419$. When applied to sequences of length m and n with $m \leqq n$, the Masek and Patterson method uses time proportional to $mn/\min(m, \log(n))$.

This content downloaded from 101.78.168.41 on Sat, 25 Nov 2017 15:20:11 UTC
All use subject to http://about.jstor.org/terms

Consider the same problem (i.e., calculating an optimal assignment) in the case that indels all have weight 1 and substitutions are either not permitted or have weight > 2 (so that it is always more economical to delete and insert than to substitute). Then it is well-known and easy to see that calculating an optimal alignment is equivalent to finding the longest common subsequence of the two sequences. Consider algorithms for the latter purpose which are restricted in the type of comparisons they can make between sequence elements. Aho, Hirschberg, and Ullman (1976) have shown that the number of comparisons of the "equal-not equal" type must be at least proportional to n^2 if the alphabet has unrestricted size, and at least proportional to ns if the alphabet has size s and $s < n$. Hirschberg (1978) has shown that the number of comparisons of the "less than-equal-greater than" type must be at least proportional to $n \log(n)$. Related problems such as finding the longest common subsequence of N sequences, finding the shortest common supersequence, and finding the longest common *consecutive* subsequence (i.e., substring) are surveyed in Hirschberg [*].

Some computational complexity results are also known for situations in which a sequence is compared not with another sequence, but with a large class, possibly infinite, of sequences defined by a "grammar". The problem is to find the grammatical sequence which is closest to the given sequence and/or to find the distance involved. A survey of computational complexity results for this problem is given by Wagner [*].

7. How Sequence Comparison Is Used

Sequence comparison can be used to answer certain clearly drawn questions, as we describe below. As with other methods, however, the most interesting current uses of sequence comparison are hard to describe in that crisp a fashion. One reason for this is that sequence comparison is involved in drawing up questions as well as in answering them. Another reason is that interesting new uses tend to spawn new methods, whose implications, potentialities, and limitations are not immediately clear.

Sequence comparison yields two kinds of information, distances and optimum analyses (e.g., alignments, traces, or listings). In some applications emphasis is on the distances, and in others on the optimum analyses. Sometimes the two kinds of applications are closely intertwined. A simple basic application emphasizing distance is a decision that two molecules (or other sequences generated by an evolutionary process) are homologous. If the distance is small enough, we may conclude that such closeness could not have occurred by chance. (However, a large distance does not imply they are nonhomologous: it may merely be that their common ancestry was very long ago.) Of course, to set the threshold requires information about how large a distance might come about by chance. This can be gained both by Monte Carlo experiments, which are often used, and by mathematical investigation (see, e.g., Deken [*] and Chvátal and Sankoff [*]). A simple basic application emphasizing optimum analysis is to reconstruct as well as possible (i.e., to estimate) the homology between two homologous molecules **a** and **b**. In other words, we want to identify which portions and which elements of **a** correspond to which portions and which elements of **b**, in the sense that corresponding portions or elements descend from a single portion or element of

This content downloaded from 101.78.168.41 on Sat, 25 Nov 2017 15:20:11 UTC
All use subject to http://about.jstor.org/terms

the ancestor molecule. This information is not only of considerable evolutionary interest, but it can also have direct biochemical value, since a portion with a particular type of activity on one molecule is likely to correspond to a portion with a similar activity on another. One way of solving this clearly drawn problem is to use the homology provided by the shortest alignment (or trace) from **a** to **b**. For reasons explained in Section 9, this is akin to maximum likelihood estimation of the homology.

Sometimes a given molecule **a** is believed not to be homologous as a whole to another molecule, but merely to contain somewhere a portion **a**′ which is homologous to a specified portion **b**′ (of some molecule **b**). Then a new problem arises of finding which portion **a**′ of **a** has the smallest distance to **b**′, what the distance between **a**′ and **b**′ is, and what the optimum (say) alignment between them is. (A similar goal is referred to as "word-spotting" in speech research.) Erickson and Sellers [*] provide two elegant and practical answers, and obtain three interesting applications. In one they discover that two plant proteins are approximately cyclical permutations of each other, a phenomenon which had not been previously observed. In another, they tentatively identify in several proteins the "translocation signal", i.e., the sequence of approximately 15 amino acids which signals that this protein is to be translocated through the endoplasmic reticulum membrane. In the third they discover that two repeating sequences of "satellite DNA" contain very similar pieces which were not previously noted. This work illustrates the fact mentioned earlier, that sequence comparison helps drawing up questions as well as answering them. If Erickson and Sellers had not been actively working with sequence comparison methods, it is not at all clear that they would have conceived the *question* described in this paragraph.

Suppose we wish to establish the homology among three molecules **a**, **b**, **c**. Since any three homologous molecules must be connected by a non-time-oriented family tree like that of Figure 13, a satisfying answer would include reconstructing

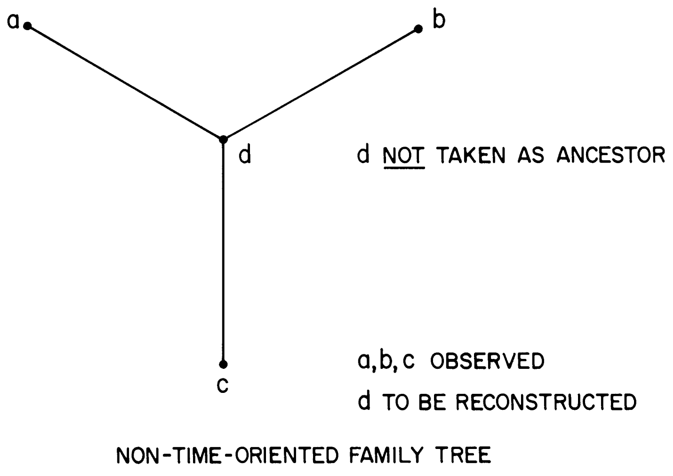


FIG. 13

This content downloaded from 101.78.168.41 on Sat, 25 Nov 2017 15:20:11 UTC
All use subject to http://about.jstor.org/terms

the internal sequence **d** as well as the homology of **d** with each of **a**, **b**, **c**, which indirectly provides the homology among **a**, **b**, **c**. (In Figure 13, the common ancestor is *not* assumed to be **d**, but may occur anywhere on the three edges. Wherever it occurs, time proceeds outward from that point along the edges of the tree. We do not concern ourselves with where the common ancestor is located.) One approach to this problem (due to Sankoff, 1975, and Sankoff et al., 1973, 1976) is to define a generalized alignment among **a**, **b**, **c**, and **d**. This means a matrix, such as

$$\begin{bmatrix} a_1 & a_2 & \phi & a_3 & a_4 \\ \phi & b_1 & b_2 & b_3 & \phi \\ c_1 & c_2 & \phi & \phi & c_3 \\ d_1 & d_2 & \phi & d_3 & d_4 \end{bmatrix},$$

in which each row consists of one of the four sequences, possibly padded by null characters ϕ, and in which no column consists entirely of null characters. The length of a generalized alignment is defined to be

$$\text{length}(\mathbf{a},\mathbf{d}) + \text{length}(\mathbf{b},\mathbf{d}) + \text{length}(\mathbf{c},\mathbf{d}) \ ,$$

where each term refers to the simple length of the ordinary two-row alignment formed by the appropriate two rows of the matrix. The generalized distance among **a**, **b**, and **c** means the minimum length, using any **d**, of any generalized alignment among **a**, **b**, **c**, and **d**. The sequence **d** and generalized alignment which provide the minimum length form a satisfying solution to the question.

An algorithm to find the optimum generalized alignment is practical though time-consuming. It is similar to the algorithm described in Section 5 above, but a three-dimensional array is used in place of the two-dimensional array. Specifically, define D_{ijk} as the generalized distance among the partial sequences $a_1 \cdots a_i$, $b_1 \cdots b_j$, and $c_1 \cdots c_k$. The recursive step to calculate D_{ijk} from the preceding values contains inner minimizations which have no counterpart in the algorithm of Section 5:

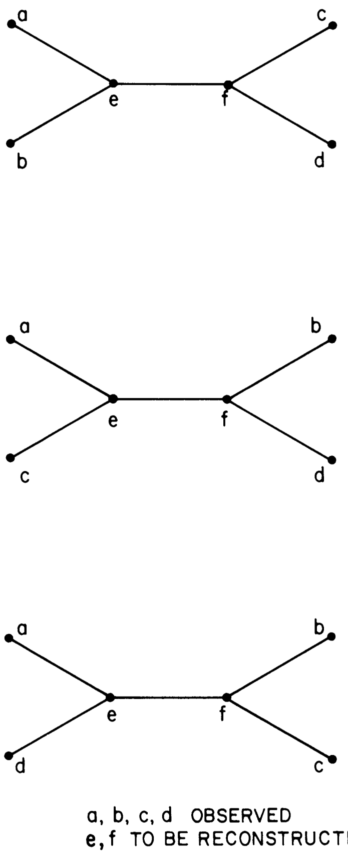
$$D_{ijk} = \min \begin{cases} D_{i-1,j-1,k-1} & + \ \min\limits_d \, [w(a_i,d)+w(b_j,d)+w(c_k,d)], \\ D_{i,j-1,k-1} & + \ \min\limits_d \, [w(\phi,d)+w(b_j,d)+w(c_k,d)], \\ D_{i-1,j,k-1} & + \quad \ldots \, , \\ D_{i-1,j-1,k} & + \quad \ldots \, , \\ D_{i,j,k-1} & + \ \min\limits_d \, [2w(\phi,d)+w(c_k,d)], \\ D_{i,j-1,k} & + \quad \ldots \, , \\ D_{i-1,j,k} & + \quad \ldots \, , \end{cases}$$

The optimum sequence **d** and the optimum generalized alignment can be found by backtracking after the recursion has been completed. An application of this method may be found in Sankoff (1973).

This content downloaded from 101.78.168.41 on Sat, 25 Nov 2017 15:20:11 UTC
All use subject to http://about.jstor.org/terms

Suppose there are more than three known homologous molecules for which we wish to obtain the corresponding information. Then the actual evolutionary history may correspond to any one of many non-time-oriented family trees, as illustrated in Figure 14 for four molecules, so it is necessary to reconstruct the tree as well as the sequences at the internal nodes and the homologies. In one approach, reconstruction of the tree is taken as a separate initial step, which we discuss in the next paragraph. Reconstruction of the internal sequences and homologies along each branch, for a known tree like that in Figure 15, is a separate subsequent step. An algorithm for this purpose was introduced in Sankoff (1975), further elucidated and applied to several sets of data in Sankoff (1976), and is most succinctly described in Sankoff and Cedergren [*]. It is a direct generalization of the algorithm just described for the three-sequence case.

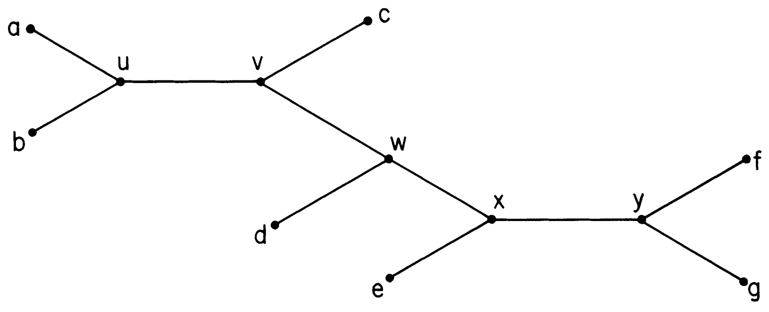
One approach to reconstruction of the non-time-oriented family tree is based on the pairwise distances among the molecules (Fitch and Margoliash, 1967 and many later papers). Of course, this tree is of great interest in itself, so reconstructing the tree is a prime application where emphasis is on the distances.



THREE DIFFERENT NON-TIME-ORIENTED FAMILY TREES

FIG. 14

This content downloaded from 101.78.168.41 on Sat, 25 Nov 2017 15:20:11 UTC
All use subject to http://about.jstor.org/terms



a, b, c, d, e, f, g OBSERVED
u, v, w, x, y TO BE RECONSTRUCTED

NON-TIME-ORIENTED FAMILY TREE

FIG. 15

Leaving molecules and moving to bird song, the pairwise distances between many songs are a tool which can be used to investigate many questions, such as the historical relationships among songs. One plausible approach would be to analyze the distances by multidimensional scaling. (Information about multidimensional scaling is available from many sources, such as Kruskal and Wish, 1978, or Carroll and Kruskal, 1978.) Bradley and Bradley [*] use another approach. Suppose there are several songs from each of several different bird populations (of a single species in a single general area). If songs change by cultural evolution, as believed, then song distances within each population should tend to be smaller than distances between populations. Bradley and Bradley [*] demonstrate that this is so, and then go on to investigate whether song distance among populations increases with geographic distance among populations, as would occur under a variety of plausible models.

Distance can also be useful in solving problems which are familiar except for the possibility of deletion-insertion or compression-expansion in addition to substitution. For example, the Acceptance-Rejection problem is to decide whether a newly observed sequence belongs to a given population. (In most applications, the characteristics of the population must be inferred from a sample of known members.) In one acceptance-rejection problem, an automatic speaker-verification system must verify whether a speaker is who she claims to be (perhaps in connection with automated banking operations over the telephone) by comparing a sample of her speech with pre-recorded samples in its memory. In another, a macromolecule must be accepted or rejected as a new member of a class of molecules which are all homologous with each other. In many such problems, including those above, distances within the population are expected to be much smaller than those between members and nonmembers, so distances provide a natural approach.

Closely related is the identification problem, which is to decide from which of several populations a newly observed sequence has been drawn. One

This content downloaded from 101.78.168.41 on Sat, 25 Nov 2017 15:20:11 UTC
All use subject to http://about.jstor.org/terms

identification problem occurs in coding theory, of course, and Levenshtein invented distances to help solve it. Here each population consists of the sequences which may be received over the noisy channel when a single sequence is transmitted. In coding theory the identification problem is referred to as decoding, and the populations generally have known probabilistic structure. Another important identification problem occurs in speech research, where one of the tasks of an automatic speech recognition system is to identify which of many words, syllables, phonemes or other units in its memory corresponds to a specific bit of speech it has isolated. Hunt, Lennig, and Mermelstein [*] consider syllable identification as one step of a larger problem, namely, identifying an entire sentence that satisfies a specified syntax. Another identification problem is to decide which grammatical computer language statement was intended in place of the ungrammatical statement which was typed in. In this application, each population consists of sequences which may result from human error when a particular sequence is intended. There is one population for each grammatical statement (in some contexts, the number of grammatical statements may not be finite). Generally distances within each population are expected to be smaller than those between populations, so distances provide a natural approach.

8. Why Should Distances Be Used?

Even though we started this paper with comparison of sequences as the goal, and with such widely-used concepts as Euclidean distance, city-block distance, and Hamming distance as the jumping-off place, it is worth reviewing the reasons for using distances. Ordinary variables, such as population or altitude (of a city), height or weight (of a person), and pitch or loudness (of a tone), characterize a *single* unit of study (such as a single city, a single person, or a single tone). Variables like distance (between two cities), perceived similarity (between two tones or other stimuli), and correlation (between two variables) characterize a *pair* of units of study. We refer to variables of the first type as *monadic* and variables of the second type as *dyadic*. Some methods of quantitative analysis explicitly require dyadic input. These include many methods of clustering (more information available from many sources, e.g., Sneath and Sokal, 1973, Wallace, 1978), and some methods for reconstructing evolutionary family trees and multidimensional scaling (see references cited above in Section 7). One reason for using distances is in order to make use of such methods. There are many potential applications of this kind using distances provided by sequence comparison, notably in the reconstruction of family trees from macromolecules.

Most methods of quantitative analysis, however, have been developed for monadic variables, which are far more common. The usual approach to quantitative study of a topic is to select several relevant monadic variables (e.g., to study human growth we might select features such as height, weight, circumference of head, length of trunk, etc.) and then to analyze these variables, by methods which range from such simple but important techniques as cross-tabulation, scatter plots, and other graphical displays, to more complicated methods, such as regression, analysis of variance, and time-series analysis.

Of course, it is of vital importance that the variables selected should not only be relevant to the topic, but also that they should cover all important aspects of the topic. For some topics, however, including the study of variable length

This content downloaded from 101.78.168.41 on Sat, 25 Nov 2017 15:20:11 UTC
All use subject to http://about.jstor.org/terms

sequences, it is difficult to find adequate monadic variables. Such topics include

(1) evolutionary study of macromolecules,

(2) study of the cultural evolution of bird song, and

(3) human perception of stimuli in many domains, such as words, faces, odors, musical compositions, etc.

Where monadic variables are inadequate, dyadic variables such as distance and similarity may provide an important new source of usable information. There are several different approaches to using the dyadic variables. One is direct common sense, illustrated for topic (1) in Erickson and Sellers [*]. Another is to adapt methods usually used with monadic variables, illustrated for topic (2) in Bradley and Bradley [*]. A third is to use methods specially intended for use with dyadic variables, like those mentioned above. Such a method may be developed or improved specifically for the intended applications. Multidimensional scaling was improved in the early 1960's specifically for use with topic (3), and there has been great growth in such applications since that time. Reconstruction of family trees from sequence comparison distances goes back at least to the 1960's, and is still undergoing development, as illustrated by Sankoff and Cedergren [*].

9. Why Levenshtein Distance?

Given that a dyadic variable may be useful, why is Levenshtein distance in particular an appropriate variable to use? We present three rationales. Our first rationale applies in a narrow range of situations, where the differences between **a** and **b** may reasonably be attributed to a series of physical changes. Most notable is the evolution of DNA and RNA, where the changes are called mutations. Another is in the comparison of computer files by programs such as DIFF in the UNIX[2] system. When several slightly different computer files of the same basic information (text, or computer source code, etc.) are in computer storage, these programs can be very helpful in keeping track of the differences, since they compare two files and describe the differences in terms of lines to be inserted, deleted, or changed, to get one version from another.

Of course, we do not know which listing from **a** to **b** is the right one, but since mutations are unlikely events a listing which is longer (even by just one elementary operation) is much more unlikely as an explanation of how **a** and **b** arose from some common ancestor than a listing which is shorter. The most plausible listing to use as an explanation is the shortest possible one, so the Levenshtein distance between two sequences is a plausible indicator of the amount of actual historical change between them. The weights which enter into the definition permit us to treat some mutations as more unlikely than others. There are sometimes convincing reasons to do this, particularly as between substitutions and indels.

The second rationale arises in many situations where it would be natural to make use of the likelihood of obtaining **b** from **a**, or of obtaining both **a** and **b** from an unspecified common ancestor. This approach has been tried, but, developing a suitable probability model for the underlying process of change can be difficult, and the calculations which result are likely to be slow and expensive.

[2]UNIX is a trademark of Bell Laboratories.

This content downloaded from 101.78.168.41 on Sat, 25 Nov 2017 15:20:11 UTC
All use subject to http://about.jstor.org/terms

The second rationale for Levenshtein distance is as an elegant, computationally practical, general-purpose approximation, not to likelihood itself but to a decreasing function of likelihood. For many uses, a monotonic function of likelihood is all that is needed.

Here is an extremely brief indication of an argument which supports the second rationale in one context. Suppose we start with a sequence **a** and permit elementary operations ("mutations") to occur by a simple Poisson-like random process. The likelihood of obtaining **b** starting from **a** is a sum of terms. Each term corresponds to one listing from **a** to **b**, and the value of that term is a product of factors. Each elementary operation in the listing contributes one factor to the product; in addition, each letter which did *not* undergo any change contributes one factor, and each position (between letters, and at the beginning and end of the sequence) at which an insertion did *not* take place contributes one factor. Under suitable assumptions, the following constitute adequate approximations:

(1) Each of the factors mentioned has a fixed value, which depends only on the type of factor, and on what letter or letters are involved in the substitution, deletion, insertion, non-changed letter, or non-inserted position.

(2) The likelihood of obtaining **b** starting from **a** is approximated by its largest term, i.e., the likelihood of one "dominant" listing.

(3) Less important, the factors which correspond to non-changed letters and to non-inserted positions are approximately equal to 1.

The dominant listing will never have more than one elementary operation for any position in the sequence. Now transform by the negative of the logarithm function. This transforms the

$$\text{approximate likelihood} \approx \text{likelihood of the dominant listing}$$

into a sum of terms $\geqq 0$, which we may call weights, i.e., into the simple length of the listing. The weights depend only on the letters involved, and if we are willing to accept approximation (3), we need only consider the weights corresponding to substitutions, deletions, and insertions, since the remaining weights are zero; not accepting (3), however, only introduces a minor complication. The approximate likelihood of obtaining **b** starting from **a** transforms into the minimum length from **a** to **b**, namely, Levenshtein distance. Thus to an approximation, distance $= -\log(\text{likelihood})$, and $-\log$ is a monotonic decreasing function.

To actually work out this rationale in detail however, is a little more involved than it appears. We have not seen this done in the literature, and our version (which we hope to publish in the future) indicates the need for assumptions and precautions which do not seem to have been stated elsewhere. For example, suppose **a** contains a run of k_1 repetitions of the same letter, and **b** contains a corresponding run of k_2 repetitions, with $m = \max(k_1,k_2)$ and $j = |k_1-k_2|$. Then the likelihood of obtaining **b** from **a** contains approximately m-choose-j largest terms of the same size, so the approximation in step (2), which is based on a single largest term being dominant, would be improved by incorporation of the factor m-choose-j. If **a** and **b** contain several pairs of corresponding runs, then the combined correction factor is the product of the individual factors for each run. This consideration is of particular importance in applications where the sequences tend to have runs, such as certain speech research applications.

This content downloaded from 101.78.168.41 on Sat, 25 Nov 2017 15:20:11 UTC
All use subject to http://about.jstor.org/terms

The third rationale is simply an application of the universal *post hoc* justification process. If we use any particular definition for distance, and find that this kind of distance supplies the information we want, that "it works" when we check its performance, then the satisfactory performance justifies the definition. Every well-made application of distance contains such checking and supports this rationale. Of course, this rationale leads naturally not only to checking on how well a given definition works but also to comparing different definitions to see which one works better. Bradley and Bradley [*] make just such a comparison, between distances based on linear time warping and Levenshtein distance, to the advantage of the latter.

REFERENCES

Aho, A. V., Hirschberg, D. S., and Ullman, J. D. (1974), *Bounds on the complexity of the longest common subsequence problem*, Journal of the Association for Computing Machinery 23,1-12.

Alberga, C. N. (1967), *String similarity and misspellings*, Communications of the ACM, 10(5), 302-313.

Auron, P. E., Rindone, W. P., Vary, C. P. H, Celentano, J. J., and Vournakis, J. N. (1982), *Computer-aided prediction of RNA secondary structures*, Nucleic Acids Research, 10(1),403-419.

Barker, W. C. Wallace, D. C. and Dayhoff, M. O. (1969), *5S Ribosomal RNA*, Atlas of Protein Sequence and Structure, Dayhoff, M. O., ed. Vol. 4, National Biomedical Research Foundation, Silver Spring, Maryland, pp. 95-98.

Bertelè, U. and Brioschi, F. (1972), *Nonserial Dynamic Programming*, Academic Press, New York.

Beyer, W. A., Stein, M. L., Smith, T. F., and Ulam, S. M. (1974), *A molecular sequence metric and evolutionary trees*, Mathematical Biosciences 19,9-25.

Bridle, J. S. (1973), *An efficient elastic-template method for detecting given words in running speech*. British Acoustical Society Spring Meeting, April 1973, Chelsea College, London, Paper 73SHC3.

Bridle, J. S. and Brown, M. D. (1979), *Connected word recognition using whole word templates*, Proceedings of the Institute for Acoustics, pp. 25-28.

Burr, D. J. (1979), *A technique for comparing curves*, Proceedings of the IEEE Conference on Pattern Recognition and Image Processing, Chicago, 1979, IEEE, New York, pp. 271-277.

Burr, D. J. (1980), *Designing a handwriting reader*, 5th International Conference on Pattern Recognition, Miami Beach, Florida, 1980, IEEE, New York, pp. 715-722.

Burr, D. J. (1981), *Elastic matching of line drawings*, IEEE Transactions on Pattern Analysis and Machine Intelligence, PAMI-3,708-713.

Calabi, L. and Hartnett, W. E. (1969a), *A family of codes for the correction of substitution and synchronization errors*, IEEE Transactions on Information Theory, IT-15(1), 102-106.

Calabi, L. and Hartnett, W. E., (1969b), *Some general results of coding theory with applications to the study of codes for the correction of synchronization errors*, Information and Control, 15,235-249.

Cannon, R. L. (1973), *An optimal text collation algorithm*, ICCH: International Conference on Computers in the Humanities, July 20-22, 1973, University of Minnesota, Final Program.

Cannon, R. L. (1976), OPCOL — *Optimal text collation algorithm*, Computers and the Humanities, 10(1),33-40.

Carroll, J. Douglas, and Kruskal, Joseph B. (1978), *Scaling, Multidimensional*, International Encyclopedia of Statistics, Kruskal, William and Tanur, Judith, M. eds., The Free Press and MacMillan, New York, pp. 892-907.

Damerau, F. J. (1964), *A technique for computer detection and correction of spelling errors*, Communications of the ACM, 7(3),171-176.

Dixon, N. R. and Martin, T. B., eds. (1979), *Automatic Speech and Speaker Recognition*, IEEE Press, New York City.

Ferguson, C. W. (1970), *Dendrochronology of bristlecone pine, Pinus aristata. Establishment of a 7484-year chronology in the White Mountains of eastern-central California, U.S.A.*, Radiocarbon Variations and Absolute Chronology, Olsson, Ingrid U., ed., Wiley Interscience, New York and Almqvist - Wiksell, Stockholm pp. 237-259

Fitch, W. M. and Margoliash, E. (1967), *Construction of phylogenetic trees*, Science, 155,279-284.

This content downloaded from 101.78.168.41 on Sat, 25 Nov 2017 15:20:11 UTC
All use subject to http://about.jstor.org/terms

Fromm, E. (1970), *An estimation of errors in the Swedish varve chronology*, Radiocarbon Variations and Absolute Chronology, Olsson, Ingrid U., ed., Wiley Interscience, New York and Almqvist - Wiksell, Stockholm, pp. 163-172.

Fujimoto, Y. et al. (1976), *Recognition of handprinted characters by nonlinear elastic matching*, 3rd International Joint Conference on Pattern Recognition, Coronado, California, 1976, IEEE, New York, pp. 113-119.

Goad, W. B. and Kanehisa, M. I. (1982), *Pattern recognition in nucleic acid sequences. I. A general method for finding local homologies and symmetries*, Nucleic Acids Research 10(1),247-263.

Gosling, J. (1981), *A redisplay algorithm*, Sigplan Notices 16(6),123-129.

Hall, P. A. V. and Dowling, G. R. (1980), *Approximate string matching*, Computing Surveys, 12(4),381-402.

Haton, J. P. (1973), *Contribution à l'analyse, paramètrisation et la reconnaissance automatique de la parole*, Thèse de doctorat d'ètat, Universitè de Nancy, Nancy, France.

Haton, Jean-Paul (1974a), *Une mèthode dynamique de comparaison de chaines de symboles de longeurs diffèrentes; application à la recherche lexicale*, Comptes Rendus Hebdomadaires des Seances de l' Academie des Sciences, Serie A 278(23),1527-1530.

Haton, J. P. (1974b), *Practical application of a real-time isolated-word recognition system using syntactic constraints*, IEEE Transactions on Acoustics, Speech and Signal Processing, ASSP-22(6),416-419.

Hirschberg, D. S. (1975), *A linear space algorithm for computing maximal common subsequences*, Communications of the ACM, 18(6),341-343.

Hirschberg, D. S. (1978), *An information theoretic lower bound for the longest common subsequence problem*, Information Processing Letters 7,40-41.

Hörnsten, Å., (1970), *Summary of discussion of the measurements and identification of varves*, Radiocarbon Variations and Absolute Chronology, Olsson, Ingrid U., ed., Wiley Interscience, New York and Almqvist - Wiksell, Stockholm, pp. 215-217.

Hunt, J. W. and McIlroy, M. D. (1976), *An algorithm for differential file comparison*, Bell Laboratories Computing Science Technical Report #41, Murray Hill, New Jersey.

Itakura, F. (1975), *Minimum prediction residual principle applied to speech recognition*, IEEE Transactions on Acoustics, Speech, and Signal Processing, ASSP-23,67-72.

Johnson, S. C. (1980), *Language-development tools on the Unix System*, Computer, 13(8),16-21.

Julesz, Bela (1971), *Foundations of Cyclopean Perception*, University of Chicago Press, Chicago.

Julesz, Bela (1978), *Global Stereopsis: Cooperative phenomena in stereoscopic depth perception*, Handbook of Sensory Physiology, Vol. 8, Perception, Held, R. et al., eds., Springer Verlag, Berlin, pp. 215-256.

Kanehisa, M. I. and Goad, W. B. (1982), *Pattern recognition in nucleic acid sequences. II. An efficient method for finding locally stable secondary structures*, Nucleic Acids Research 10(1),265-277.

Kernighan, B. W., Lesk, M. E., and Ossanna, J. F. (1978), *Document preparation*, Bell System Technical Journal 57,2115-2135.

Kruskal, Joseph B. and Wish, Myron (1978), *Multidimensional scaling*, Sage University Papers 07-011, Sage Publications, Beverly Hills and London.

Kunze, M. and Thierrin, G. (1981), *On hypercodes and error correction*, Abstract in Eleventh Conference on Numerical Mathematics and Computing, University of Manitoba, October 1981.

Larson, R. E. and Casti, J. L. (1978), *Principles of Dynamic Programming*, Marcel Dekker, New York.

Levenshtein, V. I. (1965a; English, 1966), *Binary codes capable of correcting deletions, insertions, and reversals*. Cybernetics and Control Theory 10(8),707-710, (Russian) Doklady Akademii Nauk SSSR 163(4),845-848.

Levenshtein, V. I. (1965b), *Binary codes capable of correcting spurious insertions and deletions of ones.*, Problems of Information Transmission 1(1),8-17, (Russian) Problemy Peredachi Informatsii, 1(1),12-25.

Levenshtein, V. I. (1971), *A method of constructing quasilinear codes providing synchronization in the presence of errors*, Problems of Information Transmission 7(3),215-222, (Russian) Problemy Peredachi Informatsii 7(3),30-40.

Lowrance, R. and Wagner, R. A. (1975), *An extension of the string-to-string correction problem*, Journal of the Association for Computing Machinery 22, 177-183.

MacWilliams, F. J. and Sloane, N. J. A., (1977), *Theory of Error-Correcting Codes*. North-Holland, New York.

Mills, David L. (1978), *A new algorithm to determine the Levenshtein distance between two strings*, distributed at Conference on Sequence Comparison, Universitè de Montrèal, April 1979.

Morgan, H. L. (1970), *Spelling correction in systems programs*, Communications of the ACM, 13(2),90-94.

This content downloaded from 101.78.168.41 on Sat, 25 Nov 2017 15:20:11 UTC
All use subject to http://about.jstor.org/terms

Myers, C. S. and Rabiner, L. R. (1981a), *A level building dynamic time warping algorithm for connected word recognition*, IEEE Transactions on Acoustics, Speech, and Signal Processing ASSP-29,284-297.

Myers, C. S. and Rabiner, L. R. (1981b), *Connected digit recognition using a level building DTW algorithm*, IEEE Transactions on Acoustics, Speech, and Signal Processing ASSP-29,351-363.

Myers, C. S., Rabiner, L. R., and Rosenberg, A. E. (1980), *Performance tradeoffs in dynamic time warping algorithms for isolated word recognition*, IEEE Transactions on Acoustics, Speech, and Signal Processing ASSP-28,622-635.

Needleman, Saul B. and Wunsch, Christian D. (1970), *A general method applicable to the search for similarities in the amino acid sequence of two proteins*, Journal of Molecular Biology, 48,443-453.

Norman, John M. (1975), *Elementary Dynamic Programming*, Crane, Russak, New York.

Nussinov, R., Tinoco, I. Jr., and Jacobson, A. B. (1982a), *Small changes in free energy assignments for unpaired bases do not affect predicted secondary structures in single stranded* RNA, Nucleic Acids Research 10(1),341-349.

Nussinov, R., Tinoco, I. Jr., and Jacobson, R. A. (1982b), *Secondary structure model for the complete simian virus* 40 *late precursor* RNA, Nucleic Acids Research 10(1),351-363.

Rabiner, L. R., Rosenberg, A. E., and Levinson, S. E. (1978), *Considerations in dynamic time warping for discrete word recognition*, IEEE Transactions on Acoustics, Speech, and Signal Processing ASSP-26,575-582.

Rabiner, L. R. and Schmidt, C. E. (1980), *Application of dynamic time warping to connected digit recognition*, IEEE Transactions on Acoustics, Speech, and Signal Processing, ASSP-28,337-388.

Reichert, T. A., Cohen, D. N. and Wong, A. K. C. (1973), *An application of information theory to genetic mutations and the matching of polypeptide sequences*, Journal of Theoretical Biology, 42,245-261.

Reiner, E. and Kubica, G. P. (1969), *Predictive value of pyrolysis gas-liquid chromatography in the differentiation of mycobacteria*, American Review of Respiratory Disease 99, 42-49.

Reiner E. and Bayer, Forrest L. (1978), *Botulism: a pyrolysis gas-liquid chromatographic study*, Journal of Chromatographic Science 16(12), 623-629.

Reiner, E. et al. (1979), *Characterization of normal human cells by pyrolysis gas chromatography mass spectrometry*, Biomedical Mass Spectrometry 6(11), 491-498.

Sakoe, H. and Chiba, S. (1970), *A similarity evaluation of speech patterns by dynamic programming*, Institute of Electronic Communications Engineering of Japan, July 1970,136 (in Japanese).

Sakoe, H. and Chiba, S. (1971), *A dynamic programming approach to continuous speech recognition*, 1971 Proceedings of the International Congress of Acoustics, Budapest, Hungary, Paper 20 C 13.

Sakoe, H. and Chiba, S. (1978), *Dynamic programming algorithm optimization for spoken word recognition*, IEEE Transactions for Acoustics, Speech, and Signal Processing ASSP-26,43-49.

Sankoff, David (1972), *Matching sequences under deletion/insertion constraints*, Proceedings of the National Academy of Sciences of the U.S.A. 69,4-6.

Sankoff, D. (1975), *Minimal mutation trees of sequences*, SIAM Journal on Applied Mathematics 78, 35-42.

Sankoff, D., Cedergren, R. J., and Lapalme, G. (1976), *Frequency of insertion-deletion, transversion and transition in the evolution of* 5S *ribosomal* RNA, Journal of Molecular Evolution 7, 133-149.

Sankoff, D., and Kruskal, J. B., eds. (1983), *Time-Warps, String Edit and Macromolecules: Theory and Practice of Sequence Comparison*, Addison-Wesley, Reading, Massachussetts.

Sankoff, D., Morel, C., and Cedergren, R. J. (1973), *Evolution of* 5S RNA *and the nonrandomness of base replacement*, Nature New Biology 245, 232-234.

Sellers, F. F. (1962), *Bit loss and gain correction code*, IRE Transactions on Information Theory, 8,35-38.

Smith, T. F. and Waterman, M. S. (1980), *New stratigraphic correlation techniques*, Journal of Geology, 88,451-457.

Sneath, P. H. A. and Sokal, R. R. (1973), *Numerical Taxonomy*, Freeman, San Francisco.

Tanaka, E. and Kasai, T. (1976), *Synchronization and substitution error-correcting codes for the Levenshtein metric*, IEEE Transactions on Information Theory, IT-22,156-162.

Tauber, H. (1970), *The Scandinavian varve chronology and C14 dating*, Radiocarbon Variations and Absolute Chronology, Olsson, Ingrid U., ed., Wiley Interscience, New York and Almqvist - Wiksell, Stockholm, pp. 173-196.

Ullman, J. D. (1966), *Near-optimal, single-synchronization-error-correcting code*, IEEE Transactions on Information Theory, IT-12,418-424.

Ullman, J. D. (1967), *On the capabilities of codes to correct synchronization errors*, IEEE Transactions on Information Theory, IT-13,95-105.

This content downloaded from 101.78.168.41 on Sat, 25 Nov 2017 15:20:11 UTC
All use subject to http://about.jstor.org/terms

Velichko, V. M. and Zagoruyko, N. G. (1970), *Automatic recognition of 200 words,* International Journal of Man-Machine Studies, 2,223-234.

Vintsyuk, T. K. (1968), *Speech discrimination by dynamic programming,* Cybernetics 4(1),52-57, (Russian) Kibernetika 4(1),81-88.

Wagner, Robert A. and Fischer, Michael, J. (1974), *The string-to-string correction problem,* Journal of the Association for Computing Machinery, 21(1),168-173.

Wallace, David (1978), *Clustering*, International Encyclopedia of Statistics, Kruskal, W. H. and Tanur, J. M., eds., New York, pp. 47-53.

White, G. M. and Neely, R. B. (1976), *Speech recognition experiments with linear prediction, bandpass filtering, and dynamic programming,* IEEE Transactions on Acoustics, Speech, and Signal Processing, ASSP-24,183-188.

White, G. M. (1978), *Dynamic programming, the Viterbi algorithm, and low cost speech recognition,* Proceedings of the 1978 IEEE International Conference on Accoustics, Speech, and Signal Processing, pp. 413-417.

Wong, C. K. and Chandra, A. K. (1976), *Bounds for the string editing problem*, Journal of the Association for Computing Machinery 23, 13-16.

Yasuhara, M. and Oka, M. (1977), *Signature verification experiment based on nonlinear time alignment: a feasibility study*, IEEE Transactions on Systems, Man, and Cybernetics, SMC-7, 212-216.

This content downloaded from 101.78.168.41 on Sat, 25 Nov 2017 15:20:11 UTC
All use subject to http://about.jstor.org/terms