# Significance of Nucleotide Sequence Alignments: A Method for Random Sequence Permutation That Preserves Dinucleotide and Codon Usage[1]

*Stephen F. Altschul*† and Bruce W. Erickson†*

* Department of Applied Mathematics, Massachusetts Institute of Technology
and † The Rockefeller University

The similarity of two nucleotide sequences is often expressed in terms of evolutionary distance, a measure of the amount of change needed to transform one sequence into the other. Given two sequences with a small distance between them, can their similarity be explained by their base composition alone? The nucleotide order of these sequences contributes to their similarity if the distance is much smaller than their average permutation distance, which is obtained by calculating the distances for many random permutations of these sequences. To determine whether their similarity can be explained by their dinucleotide and codon usage, random sequences must be chosen from the set of permuted sequences that preserve dinucleotide and codon usage. The problem of choosing random dinucleotide and codon-preserving permutations can be expressed in the language of graph theory as the problem of generating random Eulerian walks on a directed multigraph. An efficient algorithm for generating such walks is described. This algorithm can be used to choose random sequence permutations that preserve (1) dinucleotide usage, (2) dinucleotide and trinucleotide usage, or (3) dinucleotide and codon usage. For example, the similarity of two 60-nucleotide DNA segments from the human beta-1 interferon gene (nucleotides 196–255 and 499–558) is not just the result of their nonrandom dinucleotide and codon usage.

## Introduction

A useful measure for comparing two nucleotide sequences is *evolutionary distance,* the lowest cost for changing one sequence into the other (Sellers 1974, 1980, 1984). Costs are chosen for replacing one nucleotide by another and for inserting or deleting a nucleotide (Erickson and Sellers 1983). Each alignment of two sequences has an associated cost. The lowest-costing of all possible alignments is called the evolutionary distance. This measure has been useful for comparing nucleotide sequences from influenza viruses (Shaw et al. 1982), immunoglobulins (Litman et al. 1985), interferons (Erickson et al. 1984), and enkephalin and dynorphin (Lewis and Erickson 1985). Evolutionary distance and related measures of sequence similarity have recently been reviewed by Waterman (1984).

How small should the distance between two sequences be for their similarity to be interesting? A common practice is to permute each sequence, which involves ran-

1. Key words: interferon, graph theory, Eulerian walks, edge ordering, permutation algorithms. Abbreviations: DP = doublet preserving, DtP = doublet and triplon preserving, DTP = doublet and triplet preserving, E = edge ordering, G = graph, LU = lower-uppercase, R = reduced sequence, S = sequence, tP = triplon preserving, UL = upper-lowercase, Z = last-edge graph.

Address for correspondence and reprints: Professor Bruce W. Erickson, The Rockefeller University, 1230 York Avenue, New York, New York 10021.

*Mol. Biol. Evol.* 2(6):526–538. 1985.
© 1985 by The University of Chicago. All rights reserved.
0737-4038/85/0206-0266$02.00

domly rearranging its nucleotide order while preserving its nucleotide usage (base composition). The distance between the pair of permuted sequences is called a *permutation distance.* Calculation of a large number of permutation distances gives an estimate of the probability that any two permuted sequences would have a permutation distance less than or equal to the evolutionary distance. If this probability is small enough, it is often asserted that the similarity of the two initial sequences is the result not just of their nucleotide usage but also of their nucleotide order. Thus, the initial sequences may be related by evolutionary divergence or convergence.

This approach can be too optimistic in claiming sequence similarity. Natural nucleotide sequences are often statistically nonrandom, which can increase their similarity compared to that of artificial nucleotide sequences generated by random permutation. For instance, dinucleotide usage can differ significantly from that predicted from base composition alone (Swartz et al. 1962; Fitch 1983*a;* Lipman et al. 1984). In coding regions, the codon usage can be markedly nonrandom (Smith et al. 1983). It is important to avoid claiming that sequence similarity is the result of nucleotide order if it can be explained merely by nonrandom usage of dinucleotides and/or codons. Fitch (1983*b*) has discussed two methods that can be used to simulate nonrandom dinucleotide and codon usage. The Markov method, which is based on Markov chains, produces sequences that preserve the chosen properties of the original sequence only on the average. The permutation method chooses sequences at random among those that exactly preserve the chosen properties. As noted by Fitch (1983*b*), the permutation method *requires* but the Markov method only *expects* a random sequence to preserve the chosen properties. This paper describes and illustrates a method that generates with equal probability all permutations with a given dinucleotide usage or dinucleotide and codon usage.

Fitch (1983*b*) states that for two sequences of sufficient length (~300 residues for dinucleotide usage), the two methods should be equivalent. For two interferon DNA sequences of length 60 bp described below, we found that the Markov and permutation methods yielded notably different distributions of distances. A sample of 1,000 pairs of sequences that preserved dinucleotide usage was generated. The sample mean and standard deviation (SD), respectively, were 41.74 and 3.86 for the Markov method but 39.24 and 2.93 for the permutation method. The range of distances was broader for the Markov method (27–55) than for the permutation method (30–46). Fitch (1983*b*) has discussed the appropriateness of using these methods in different contexts.

## Terminology

Although their formulation is specifically motivated by their capacity for generating permutations of nucleotide sequences, the following algorithms can be applied to sequences of any kind. They are based on a theorem first proved by van Aardenne-Ehrenfest and de Bruijn (1951) and later restated by Kasteleyn (1967, p. 77), Knuth (1973, p. 375), and Zaman (1984, p. 225). The statement and proof of this theorem presented below contain as little mathematical terminology as possible in order to make the operation of the algorithms comprehensible to the general reader.

We shall use the term *singlet* rather than mononucleotide, *doublet* in place of dinucleotide, and *triplet* instead of trinucleotide. Rather than speak of codons, we define a *triplon* as being a member of a set of consecutive nonoverlapping triplets. These terms are illustrated in figure 1.

```
       1   5    10    15    20    25
S1 = AGACATAAAGTTCCGTACTGCCGGGAT

S2 = AAGTTACGAATACATCCCTGGAGGCGT  (DP)

S3 = AGTACTGCCGTTCCGGGATAAAGACAT  (DTP)

S4 = AAAGATCCGGTTAGACGGTACTGCCAT  (DtP)
```

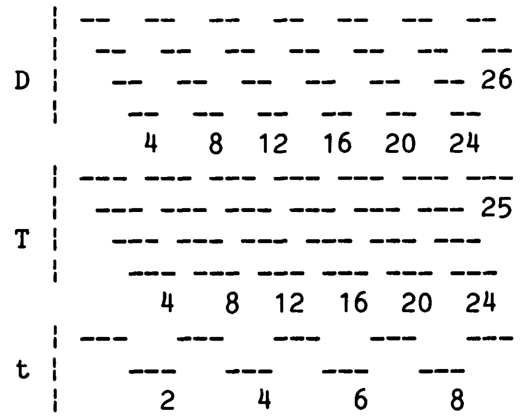

FIG. 1.—Sequence $S_1$ and three permutations of $S_1$. The set of doublets (D), triplets (T), and one set of triplons (t) for $S_1$ are illustrated. $S_2$ is a doublet-preserving (DP) permutation, $S_3$ is a doublet-and-triplet-preserving (DTP) permutation and $S_4$ is a doublet-and-triplon-preserving (DtP) permutation.

By definition, sequence permutation preserves singlet usage and sequence length. Generation of a random triplon-preserving (tP) permutation is easy, since it involves only random permutation of one set of nonoverlapping triplons. Generation of a random doublet-preserving (DP), doublet-and-triplet-preserving (DTP), or doublet-and-triplon-preserving (DtP) permutation is more difficult because the elements to be preserved overlap. Fitch (1983*b*) has described an algorithm for generating DP permutations, but it does not generate all permutations with equal probability.[2] We describe a modification of Fitch's algorithm that does generate random DP permutations. Lipman et al. (1984) mention without details that they have used such an algorithm. We show how our algorithm can be extended to generation of random DTP or DtP permutations.

As Fitch (1983*b*) has noted, generating a random DP permutation is equivalent to finding a random Eulerian walk in a directed multigraph. This statement can be explained by using the following terms. A *directed graph* is a set of *vertices* (points) and a set of *directed edges* (edges, arrows) between certain pairs of vertices. An edge from vertex $x$ to vertex $y$ is both an *edge from* $x$ and an *edge to* $y$. A *loop* is an edge from a vertex to the same vertex. A *multigraph* is a graph that can have two or more

2. For example, there are two DP permutations of the sequence AATAT, the original sequence and ATAAT. Fitch's algorithm generates the original sequence with probability 1/3 and the second sequence with probability 2/3. If one adds the extra edge TA to the doublet graph as Fitch suggests, his method still generates ATAAT twice as frequently as it does AATAT.

edges from $x$ to $y$. An *ordered graph* is a graph whose edges are ordered in a sequence from the first to the last. A *subgraph* of a graph G consists of the set of vertices in G and a subset of the edges in G. A *walk* in a graph is a sequence of vertices $x_1\ x_2 \cdots x_n$ such that every pair of consecutive vertices $x_ix_{i+1}$ is joined by an edge from $x_i$ to $x_{i+1}$. A vertex can be present in a walk more than once. An *Eulerian walk* is a walk that uses each edge in the graph exactly once. Vertex $x_1$ is connected to vertex $x_2$ in a graph G if there is a walk from $x_1$ to $x_2$ in G. Note that $x_1$ may be connected to $x_2$ even though $x_2$ is not connected to $x_1$. Every vertex is connected to itself.

## Random Doublet-preserving Permutation

### Graphs and Edge Orderings

Given a sequence $S = s_1\ s_2 \cdots s_f$, construct a *doublet graph* G that has a vertex for each singlet that appears somewhere in $S$ and one edge from vertex $s_i$ to vertex $s_{i+1}$ for each occurrence of the doublet $s_is_{i+1}$ in $S$. The *s edge list* is an ordered list of all edges from vertex $s$. An *edge ordering* E of the graph is a complete set of edge lists for G. Note that the sequence $S$ determines an edge ordering E($S$) as well as the graph G, because the doublets beginning at $s_1s_2$ occur in a specific order in $S$. For example, DNA sequence $S_1$ of figure 1 specifies the graph G($S_1$) of figure 2 and determines edge ordering E($S_1$) of figure 3. As in this example, the doublet graph of a sequence is a directed multigraph that may contain loops.

Just as $S$ uniquely determines an edge ordering E($S$) of its graph, so conversely this edge ordering uniquely determines $S$. Furthermore, E($S$) uniquely determines an Eulerian walk in G from $s_1$ in the obvious way: follow the first edge in the $s_1$ edge list to $s_2$, strike this edge from the list, follow the first edge of the $s_2$ edge list to $s_3$, and so forth. For this reason, E($S$) will be called an *Eulerian edge ordering.* On the left of figure 4, the edges of G are numbered according to the edge ordering E($S_1$). This numbering can be interpreted as an Eulerian walk in G.

If sequence $S'$ is a doublet-preserving permutation of $S$, it determines the same doublet graph as $S$ because each doublet of $S'$ corresponds to a doublet of $S$, which in turn corresponds to an edge of G. The edge orderings E($S$) and E($S'$) differ only in the internal order of their edge lists. For example, sequence $S_2$ of figure 1 is a DP permutation of $S_1$. Its edge ordering E($S_2$) is shown in figure 3 as a set of edge lists and in the center of figure 4 as the edge numbering of an ordered graph, which can

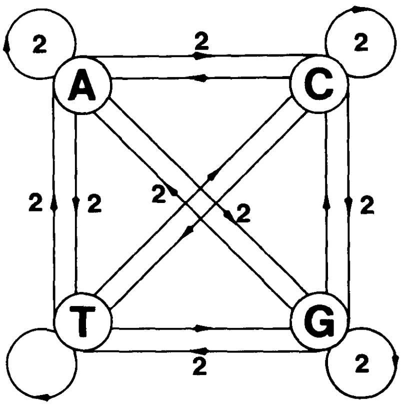

FIG. 2.—The doublet graph G($S_1$). Ten edges are used twice, as indicated.

| Edge list | E(S1) | E(S2) | E* |
|---|---|---|---|
| | 1 3 5 7 8 9 17 26 | 1 2 6 9 10 12 14 22 | 1 2 11 13 20 23 |
| A: | AG AC AT AA AA AG AC AT | AA AG AC AA AT AC AT AG | AA AG AT AC AC AT [AA AG] |
| | 4 13 14 18 21 22 | 7 13 16 17 18 25 | 4 9 14 15 21 22 |
| C: | CA CC CG CT CC CG | CG CA CC CC CT CG | CT CG CC CG CC CA |
| | 2 10 15 20 23 24 25 | 3 8 20 21 23 24 26 | 3 6 7 10 16 17 |
| G: | GA GT GT GC GG GG GA | GT GA GG GA GG GC GT | GC GG GT GA GG GT [GA] |
| | 6 11 12 16 19 | 4 5 11 15 19 | 5 8 12 18 19 |
| T: | TA TT TC TA TG | TT TA TA TC TG | TG TC TA TT TA |

FIG. 3.—The edge orderings E($S_1$), E($S_2$), and E*. In each edge ordering, numbers indicate the order of each edge in the corresponding long walk. Edges in brackets are not used in the long walk. Underlined edges belong to the last-edge graph determined by the ordering.

be interpreted as an Eulerian walk on G. In general, each DP permutation $S'$ of $S$ preserves the terminal singlets $s_1$ and $s_f$ of $S$ and specifies a unique Eulerian edge ordering of G. Conversely, each Eulerian walk in G from $s_1$ to $s_f$ specifies a unique DP permutation of $S$.

A new edge ordering of G can be generated by separately and randomly permuting each edge list of E($S$). This random edge ordering determines a *long walk* in G that starts at vertex $s_1$ and ends at a vertex whose edge list has been exhausted. This vertex must be $s_f$, the final vertex of $S$. Since such a long walk in G usually ends before every edge is used, it is seldom an Eulerian walk. For example, consider the edge ordering E* on the right of figure 3. Its long walk on the right of figure 4 is not Eulerian because three edges (AA, AG, GA) are still not used when final vertex T is reached for the last time. It is inefficient to trace a long walk most of the way through G before finding that it is not Eulerian. The following theorem provides a general criterion for quickly determining whether or not an edge ordering of G is Eulerian. Given an edge ordering E of G, let the *last edge* from vertex $x$ be the final edge of the $x$ edge list. The *last-edge graph* Z is the subgraph of G consisting of all last edges except that of the final vertex $s_f$.

## Eulerian Edge-ordering Theorem

An edge ordering E is Eulerian if and only if all vertices in the last-edge graph Z are connected in Z to $s_f$. We prove this theorem in two parts.

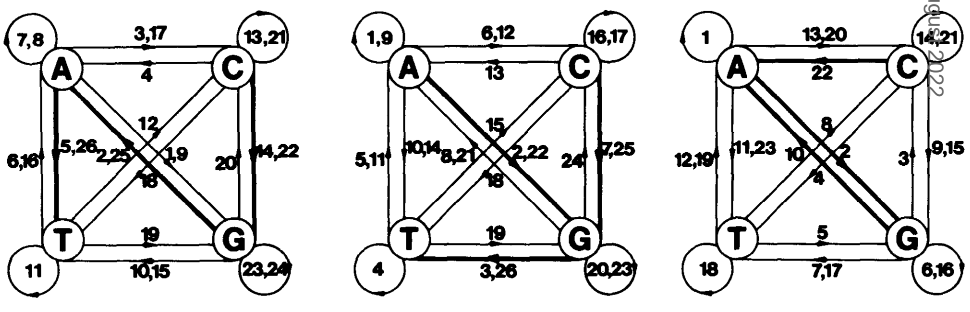

FIG. 4.—Three long walks. *Left,* the Eulerian long walk determined by E($S_1$). *Center,* the Eulerian long walk determined by E($S_2$). *Right,* the non-Eulerian long walk determined by E*. The emphasized edges belong to the last-edge graph determined by the corresponding edge ordering.

### *Part A: If E is Eulerian, then All Vertices Are Connected in Z to $s_f$*

A vertex is *exhausted* during a walk in G when its last edge is used. Note that when a vertex other than $s_f$ is exhausted, not only have all edges from it been used but also all edges to it. If E is Eulerian, the long walk determined by E exhausts all vertices in G. Except for $s_f$, number these vertices from 1 to $N$ in the order in which they are exhausted. The last edge from $N$ must point to $s_f$ because all long walks must end at $s_f$ and all other vertices have been exhausted. Thus vertex $N$ is connected in Z to $s_f$. The last edge from $N - 1$ must point to $N$ or $s_f$ because vertices $1, \cdots N - 1$ have been exhausted; thus $N - 1$ is also connected in Z to $s_f$. By similar reasoning, every vertex is connected in Z to $s_f$.

### *Part B: If E Is Not Eulerian, then Not All Vertices Are Connected in Z to $s_f$*

Let $U$ be the set of all vertices not exhausted during the long walk determined by E. Since E is not Eulerian, $U$ must contain at least one vertex because the long walk determined by E does not use all edges of G. Vertex $s_f$ is not a member of $U$ because $s_f$ is exhausted in all long walks. Each edge not used in the walk points to a vertex in $U$ because all other vertices are exhausted. In particular, the last edge of each vertex in $U$ points to a vertex in $U$. Thus all walks in Z that begin at a vertex in $U$ must end at a vertex in $U$. Therefore, no vertex in $U$ can be connected in Z to $s_f$. This completes the proof.

## Random Doublet-preserving Permutation Algorithm

A random doublet-preserving permutation $S'$ of sequence $S$ is generated by following steps (1) to (6).

(1) Construct the doublet graph G and edge ordering E corresponding to $S$.

(2) For each vertex $s$ in G except $s_f$, randomly select one edge from the $s$ edge list of E($S$) to be the last edge of the $s$ list in a new edge ordering.

(3) From this set of last edges, construct the last-edge graph Z and determine whether or not all of its vertices are connected to $s_f$.

(4) If any vertex is not connected in Z to $s_f$, the new edge ordering will not be Eulerian, so return to (2). If all vertices are connected in Z to $s_f$, the new edge ordering will be Eulerian, so continue to (5).

(5) For each vertex $s$ in G, randomly permute the remaining edges of the $s$ edge list of E($S$) to generate the $s$ edge list of the new edge ordering E($S'$).

(6) Construct sequence $S'$, a random DP permutation of $S$, from E($S'$) as follows. Start at the $s_1$ edge list. At each $s_i$ edge list, add $s_i$ to $S'$, delete the first edge $s_is_j$ of the edge list, and move to the $s_j$ edge list. Continue this process until all edge lists are exhausted.

By the Eulerian edge-ordering theorem, E($S'$) is Eulerian because all vertices of its last-edge graph are connected to $s_f$. Sequence $S'$ is a DP permutation of $S$ because by construction both $S$ and $S'$ specify Eulerian edge orderings of the same graph G. Finally, $S'$ is a random DP permutation because edge ordering E($S'$) was randomly selected from the set of all Eulerian edge orderings.

## DNA Examples

This algorithm is efficient because only one edge must be chosen from each edge list except the $s_f$ edge list in order to determine whether or not a DP permutation will result. In particular, the Eulerian edge-ordering theorem guarantees that for any DNA sequence no more than three last edges (doublets) need to be selected before deciding whether or not a DP permutation will result. For example, generation of the random DP permutation $S_2$ from $S_1$ is illustrated in figure 3. AG was randomly selected from the eight doublets of the A edge list to be a new last edge. Similarly, CG was selected from the C edge list and GT from the G edge list. As shown in the middle graph of figure 4, all vertices of the emphasized last-edge graph are connected to T, so E($S_2$) is Eulerian. The last edge of the T edge list is not present in Z because T is the final singlet of $S_1$. In contrast, as shown by the graph on the right in figure 4, three vertices of this emphasized last-edge graph are not connected to T, so this edge ordering of G($S_1$) is not Eulerian.

The Eulerian edge-ordering theorem can be used to calculate not only the probability ($P$) that a random edge ordering of a graph G will be Eulerian but also the number of possible Eulerian walks starting at $s_1$ (van Aardenne-Ehrenfest and de Bruijn 1951). Although $P$ depends on graph G, an approximate value of $P$ can be calculated by assuming that the last edge from each vertex is equally likely to point to any vertex. For a long DNA sequence $S$ containing all four nucleotides, $P = \sim 1/4$. In other words, ~3/4 of the random edge orderings of G are not Eulerian. Each of these undesired edge orderings is efficiently rejected as soon as one vertex of its last-edge graph is known to be not connected to $s_f$.

## Random Doublet-and-Triplet-preserving Permutation

The DP permutation algorithm can be modified to generate a random permutation of sequence $S$ that preserves not only singlet and doublet usage but also triplet usage. The *triplet graph* G′ of $S$ is the graph having a vertex for each doublet that appears somewhere in $S$ and an edge from $s_is_{i+1}$ to $s_{i+1}s_{i+2}$ for each occurrence of the triplet $s_is_{i+1}s_{i+2}$ in $S$. $S$ uniquely defines an Eulerian walk in G′ from $s_1s_2$ to $s_{f-1}s_f$ and vice versa. A random Eulerian walk in G′ starting at $s_1s_2$ corresponds to a random DTP permutation of $S$.

The Eulerian edge-ordering theorem can be applied to G′ just as it was to the doublet graph G. The random DP permutation algorithm is readily extended to an algorithm for generating a random DTP permutation of sequence $S$. For instance, DNA sequence $S_1$ specifies an edge ordering E′($S_1$) that consists of 16 edge lists (fig. 5). Edge ordering E′($S_3$) was generated by separately and randomly permuting these lists. The 15 edges of Z($S_3$) are underlined. All 16 doublet vertices in Z($S_3$) are connected to AT, so edge ordering E′($S_3$) is Eulerian and sequence $S_3$ of figure 1 is a random DTP permutation of $S_1$.

These algorithms can be extended to a set of algorithms for finding random permutations that preserve all doublets, triplets, · · · $n$-tuplets of sequence $S$. As the length of the largest preserved $n$-tuplet increases, the number of possible permutations decreases, until at some point only $S$ is possible.

## Random Doublet-and-Triplon-preserving Permutation

Both dinucleotide usage and codon usage of coding DNA sequences are often nonrandom. The appropriate permutation method for this situation should preserve both the doublet usage and usage of just one of the three sets of triplons. Consider a sequence $S$ whose length is divisible by three and whose first triplon is $t_1 = s_1s_2s_3$, as illustrated in figure 1. Since triplon $s_is_{i+1}s_{i+2}$ contains doublets $s_is_{i+1}$ and $s_{i+1}s_{i+2}$, preservation of a triplon also preserves both intratriplon doublets. Thus, the problem of preserving the triplons and doublets is reduced to the problem of preserving the triplons

| Edge list | $E'(S_1)$ | $E'(S_3)$ | Edge list | $E'(S_1)$ | $E'(S_3)$ |
|---|---|---|---|---|---|
| | 7 8 | 20 21 | | 2 25 | 17 23 |
| AA: | AAA AAG | AAA AAG | GA: | GAC GAT | GAT GAC |
| | 3 17 | 4 24 | | 20 | 7 |
| AC: | ACA ACT | ACT ACA | GC: | GCC | GCC |
| | 1 9 | 1 22 | | 23 24 | 15 16 |
| AG: | AGA AGT | AGT AGA | GG: | GGG GGA | GGG GGA |
| | 5 | 18 | | 10 15 | 2 10 |
| AT: | ATA | ATA | GT: | GTT GTA | GTA GTT |
| | 4 | 25 | | 6 16 | 3 19 |
| CA: | CAT | CAT | TA: | TAA TAC | TAC TAA |
| | 13 21 | 8 13 | | 12 | 12 |
| CC: | CCG CCG | CCG CCG | TC: | TCC | TCC |
| | 14 22 | 9 14 | | 19 | 6 |
| CG: | CGT CGG | CGT CGG | TG: | TGC | TGC |
| | 18 | 5 | | 11 | 11 |
| CT: | CTG | CTG | TT: | TTC | TTC |

FIG. 5.—Edge orderings $E'(S_1)$ and $E'(S_3)$ for the triplet graph $G'(S_1)$.

and intertriplon doublets, which is solved by using the random DP permutation algorithm.

## Random Doublet-and-Triplon-preserving Permutation Algorithm

A random doublet-and-triplon-preserving permutation $S'$ of sequence $S$ is generated by following steps (1) to (6).

(1) Represent sequence $S$ as a sequence of uppercase letters.

(2) Change the uppercase letters at positions 3, 6, 9, · · · $n$ into the corresponding lowercase letters to generate sequence $S^*$.

(3) Assign each triplon to an ordered triplon list according to its first and last letters and its order in $S^*$. For example, AGa and ATa are both stored in the Aa triplon list.

(4) Delete the letters at positions 2,5,8, · · · $n - 1$ from $S^*$ to form a reduced sequence $R$, which contains alternating upper-lowercase (UL) doublets and lower-uppercase (LU) doublets.

(5) Treating the upper- and lowercase letters as distinct, generate from sequence $R$ a random DP permutation $R'$ by using the algorithm described above.

(6) Expand random DP permutation $R'$ into the random DtP permutation $S'$ as follows. Randomly permute each triplon list. Start at the first UL doublet of $R'$. Replace the current UL doublet in $R'$ with the first triplon of the corresponding triplon list, delete this triplon from the list, and move to the next UL doublet in $R'$. Continue this process until the triplon lists are exhausted. Change each lowercase letter back into the corresponding uppercase letter to produce $S'$.

During step (5), the UL doublets correspond to triplons in $S$ and the LU doublets correspond to intertriplon doublets in $S$. The random DP permutation $R'$ preserves the alternation of these distinct sets of doublets. The intratriplon doublets are stored during step (3), removed in step (4), and replaced during step (6). Since each triplon list is randomly permuted before $R'$ is expanded, sequence $S'$ is a random DtP permutation of $S$. This algorithm is readily extended to a set of algorithms that preserve all doublets and one set of $n$-tuplons, where an $n$-tuplon is a member of a set of contiguous nonoverlapping $n$-tuplets.

## A DNA Example

The random DtP permutation algorithm is illustrated by the generation from DNA sequence $S_1$ of the new sequence $S_4$, which preserves all eight triplons (codons) and 23 doublets (dinucleotides) of $S_1$. First, $S_1$ is converted into equivalent sequence $S_1^*$ by changing A to a at positions 3 and 9, C to c at positions 18 and 21, G to g at positions 15 and 24, and T to t at positions 6, 12, and 27, as shown in figure 6. Next, the nine triplons are stored in the five triplon lists of $S_1^*$ shown in figure 7. Then reduced sequence $R_1$ is generated by deleting the middle singlet of each triplon. $R_1$ contains 18 singlets of eight types (A, C, G, T, a, c, g, t) and 17 doublets, which represent five of the 16 possible UL doublet types and eight of the 16 possible LU doublet types.

Next, edge ordering E($R_1$) is constructed and seven doublets are randomly selected, one from each edge list except the t list, as shown in figure 7. The eight vertices of the last-edge graph having these seven doublets as its edge set are connected to final vertex t, as shown at the top of figure 8. Thus a new edge ordering having these last edges will be Eulerian. One such edge ordering, E($R_4$) of figure 7, is randomly selected by completing the random permutation of each list of E($R_1$). This edge ordering corresponds to the Eulerian long walk at the bottom of figure 8. Then random DP permutation $R_4$ is expanded into sequence $S_4^*$ of figure 6 by replacing each UL doublet of $R_4$ by the next unused triplon from the corresponding permuted triplon list. Finally, the eight lowercase letters are changed back into uppercase letters to generate DNA sequence $S_4$, which is a random DtP permutation of DNA sequence $S_1$.

## Significance of an Interferon Alignment

A practical application of these permutation algorithms is embodied in the significance of the similarity of two sequences from the human gene for beta-1 interferon (Erickson et al. 1984). The DNA sequence of this gene was determined by Ohno and Taniguchi (1981) and by Fiers et al. (1982). Using a substitution cost of one and an insertion/deletion cost of two, the evolutionary distance between sequence A (nucleotides 196–255) and sequence B (nucleotides 499–558) is 25. The alignment having this distance is shown in figure 9. Can this distance be explained by base composition alone? To answer this question, each sequence was permuted 1,000 times, and 1,000 permutation distances were calculated. A histogram of these distances is shown at the top of figure 10. The distance of 25 is 6.0 SDs less than the mean permutation distance. If the distribution of permutation distances is assumed to be Gaussian, this is a highly significant result. In fact, the distribution is somewhat non-Gaussian. If one wishes to

```
          3     6     9    12    15    18    21    24    27
S1  = A G A C A T A A A G T T C C G T A C T G C C G G G A T

        2     5     8    11    14    17    20    23    26
S1* = A G a C A t A A a G T t C C g T A c T G c C G g G A t

      1     3     5     7     9    11    13    15    17  18
R1  = A   a C   t A   a G   t C   g T   c T   c C   g G   t

      1     3     5     7     9    11    13    15    17  18
R4  = A   a G   t C   g G   t A   a C   g T   c T   c C   t

        2     5     8    11    14    17    20    23    26
S4* = A A a G A t C C g G T t A G a C G g T A c T G c C A t

          3     6     9    12    15    18    21    24    27
S4  = A A A G A T C C G G T T A G A C G G T A C T G C C A T
```

FIG. 6.—A series of sequences constructed during generation of $S_4$, a DtP permutation of $S_1$.

| Edge list | $E(R_1)$ | $E(R_4)$ | Triplon list | $S_1^*$ | $S_4^*$ |
|---|---|---|---|---|---|
| | 1 5 | 1 9 | | 1 3 | 1 5 |
| A: | Aa $\underline{\text{Aa}}$ | Aa $\underline{\text{Aa}}$ | AXa: | AGa AAa | AAa AGa |
| | 3 9 15 | 5 11 17 | | 5 8 | 3 6 |
| C: | Ct Cg $\underline{\text{Cg}}$ | Cg Cg $\underline{\text{Ct}}$ | CXg: | CCg CGg | CCg CGg |
| | 7 17 | 3 7 | | 2 | 9 |
| G: | Gt $\underline{\text{Gt}}$ | Gt $\underline{\text{Gt}}$ | CXt: | CAt | CAt |
| | 11 13 | 13 15 | | 4 9 | 2 4 |
| T: | Tc $\underline{\text{Tc}}$ | Tc $\underline{\text{Tc}}$ | GXt: | GTt GAt | GAt GTt |
| | 2 6 | 2 10 | | 6 7 | 7 8 |
| a: | aC $\underline{\text{aG}}$ | aG $\underline{\text{aC}}$ | TXc: | TAc TGc | TAc TGc |
| | 12 14 | 14 16 | | | |
| c: | cT $\underline{\text{cC}}$ | cT $\underline{\text{cC}}$ | | | |
| | 10 16 | 6 12 | | | |
| g: | gT $\underline{\text{gG}}$ | gG $\underline{\text{gT}}$ | | | |
| | 4 8 | 4 8 | | | |
| t: | tA tC | tC tA | | | |

FIG. 7.—Edge orderings $E(R_1)$ and $E(R_4)$ and triplon lists for $S_1^*$ and $S_4^*$.

make no assumptions about tail behavior, order statistics can be used to assess the significance of the result. If $N$ permutation distances are calculated and $M$ of the $N$ are as small or smaller than the evolutionary distance in question, then the probability that the distance is due to chance is $(M + 1)/(N + 1)$. In the present case, order statistics allow one to reject with 99.9% confidence the null hypothesis that the evolutionary distance between A and B is explainable by the similarity of their base compositions alone.

The algorithms described above were used to permute DNA sequences A and B while preserving their codon usage, dinucleotide usage, or both simultaneously. Histograms representing 1,000 tP, DP, and DtP permutation distances are shown in figure 10. The evolutionary distance of 25 is respectively 5.3, 4.9, and 4.7 SDs less than the mean permutation distance. The DP and DtP distributions are definitely non-Gaussian, so it is best to use order statistics when assessing the significance of the evolutionary distance. The hypothesis that the distance between A and B can be explained by the

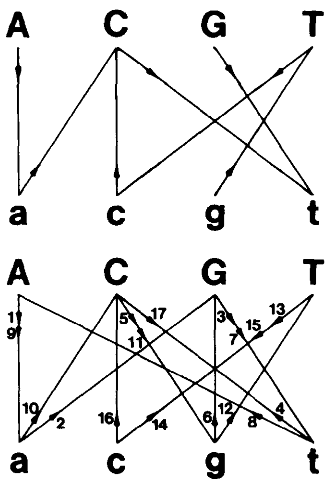

FIG. 8.—Last-edge graph $Z(R_4)$ and the Eulerian long walk in $G(R_4)$ determined by $E(R_4)$.

```
196 200        210        220        230        240        250  255
  CTCCTGTGGCAATTGAATGGGAGGCTTGAATACTGCCTCAAGGACAGGATGAACTTTGAC
  C CCTG     AT   ATGGGAGG TT    A T CCT AAGG CA G  G AC  T AC
  CACCTGAAAAGATATTATGGGAGGATTCTGCATTACCTGAAGGCCAAGGAGTACAGTCAC
499           510        520        530        540        550     558
```

FIG. 9.—Alignment of sequences A and B from the human beta-1 interferon gene. The center line echoes the 35 nucleotide identities.

similarity of their dinucleotide and codon usage alone can be rejected with 99.9% confidence.

In this example, the small distance between the beta-1 interferon sequences A and B is better explained by the order of their nucleotides (Erickson et al. 1984). In other cases, the dinucleotide and codon usage may be sufficient to explain an evolutionary distance that would seem significant if only base composition were considered. The permutation algorithms described in this paper are useful in recognizing such cases.

For example, let sequence C be the same as B except that it begins CACCT-*T*AAAAGATAT*GGG*G*A*GAG · · ·, where the italicized elements are different from

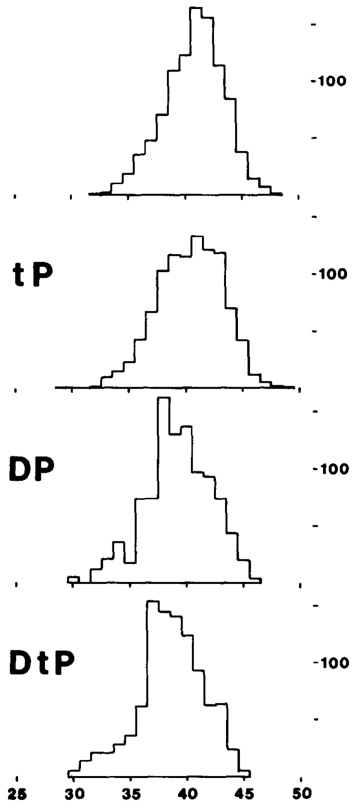

FIG. 10.—Four histograms of 1,000 permutation distances for sequences A and B from the human beta-1 interferon gene. *Top,* simple permutations; *tP,* triplon-preserving permutations; *DP,* doublet-preserving permutations; *DtP,* doublet-and-triplon-preserving permutations.

B. The evolutionary distance between A and C is 30. Sequence C was chosen to have the same singlet and doublet usage as B, so the DP histogram of figure 10 also applies to the comparison of A and C. Since five of these permutation distances are $\leqslant 30$, the hypothesis that the distance of 30 between A and C can be explained by the similarity of their dinucleotide usage alone can*not* be rejected with 99.5% confidence. But from the top histogram of figure 10, the hypothesis that the distance of 30 can be explained by base composition alone can still be rejected with 99.9% confidence.

## Acknowledgments

We thank Professor Walter M. Fitch (University of Wisconsin) for suggesting this problem. This work was supported by Public Health Service Grant GM 32622 from the National Institute of General Medical Sciences, Department of Health and Human Services, and by a contract from the Office of Naval Research and the Army Research Office.

## LITERATURE CITED

ERICKSON, B. W., L. T. MAY, and P. B. SEHGAL. 1984. Internal duplication in human alpha-1 and beta-1 interferons. Proc. Natl. Acad. Sci. USA **81**:7171–7175.

ERICKSON, B. W., and P. H. SELLERS. 1983. Recognition of patterns in genetic sequences. Pp. 55–91 *in* D. SANKOFF and J. B. KRUSKAL, eds. Time warps, string edits, and macromolecules: the theory and practice of sequence comparison. Addison-Wesley, Reading, Mass.

FIERS, W., E. REMAUT, R. DEVOS, H. CHEROUTRE, R. CONTRERAS, D. GHEYSON, W. DEGRAVE, P. STANSSENS, J. TAVERNIER, Y. TAYA, and J. CONTENT. 1982. The human fibroblast and human immune interferon genes and their expression in homologous and heterologous cells. Phil. Trans. R. Soc. Lond. **B299**:29–38.

FITCH, W. M. 1983*a*. Calculating the expected frequencies of potential secondary structure in nucleic acids as a function of stem length, loop size, base composition and nearest-neighbor frequencies. Nucleic Acids Res. **11**:4655–4663.

———. 1983*b*. Random sequences. J. Mol. Biol. **163**:171–176.

KASTELEYN, P. W. 1967. Graph theory and crystal physics. Pp. 43–110 *in* F. HARARY, ed. Graph theory and theoretical physics. Academic Press, London.

KNUTH, D. E. 1973. Fundamental algorithms. Addison-Wesley, Reading, Mass.

LEWIS, R. V., and B. W. ERICKSON. 1985. Evolution of proenkephalin and prodynorphin. Am. Zool. (accepted).

LIPMAN, D. J., W. J. WILBUR, T. F. SMITH, and M. S. WATERMAN. 1984. On the statistical significance of nucleic acid similarities. Nucleic Acids Res. **12**:215–226.

LITMAN, G. W., K. MURPHY, L. BERGER, R. LITMAN, K. HINDS, and B. W. ERICKSON. 1985. Complete nucleotide sequences of three $V_H$ genes in *Caiman*, a phylogenetically ancient reptile: evolutionary diversification in coding segments and variation in the structure and organization of recombination elements. Proc. Natl. Acad. Sci. USA **82**:844–848.

OHNO, S., and T. TANIGUCHI. 1981. Structure of a chromosomal gene for human interferon beta. Proc. Natl. Acad. Sci. USA **78**:5305–5309.

SELLERS, P. H. 1974. On the theory and computation of evolutionary distances. SIAM J. Appl. Math. **26**:787–793.

———. 1980. The theory and computation of evolutionary distances: pattern recognition. J. Algorithms **1**:359–373.

———. 1984. Pattern recognition in genetic sequences by mismatch density. Bull. Math. Biol. **46**:501–514.

SHAW, M. W., R. A. LAMB, D. J. BRIEDIS, B. W. ERICKSON, and P. W. CHOPPIN. 1982. Complete nucleotide sequence of the neuraminidase gene of influenza B virus. Proc. Natl. Acad. Sci. USA **79**:6817–6821.

SMITH, T. F., M. S. WATERMAN, and J. R. SADLER. 1983. Statistical characterization of nucleic acid sequence functional domains. Nucleic Acids Res. **11**:2205–2220.

SWARTZ, M. N., T. A. TRAUTNER, and A. KORNBERG. 1962. Enzymatic synthesis of deoxyribonucleic acid. J. Biol. Chem. **237**:1961–1967.

VAN AARDENNE-EHRENFEST, T., and N. G. DE BRUIJN. 1951. Circuits and trees in oriented linear graphs. Simon Stevin **28**:203–217.

WATERMAN, M. S. 1984. General methods of sequence comparison. Bull. Math. Biol. **46**:473–500.

ZAMAN, A. 1984. Urn models for Markov exchangeability. Ann. Probability **11**:223–229.

WALTER M. FITCH, reviewing editor

Received February 19, 1985; revision received June 10, 1985.