CABIOS Vol. 13 no. 1 1997
Pages 75-80

A tool for aligning very similar DNA sequences

Kun-Mao Chao, Jinghui Zhang[1], James Ostell[1] and Webb Miller[2,3]

Abstract

Results: *We have produced a computer program, named sim3, that solves the following computational problem. Two DNA sequences are given, where the shorter sequence is very similar to some contiguous region of the longer sequence. Sim3 determines such a similar region of the longer sequence, and then computes an optimal set of single-nucleotide changes (i.e. insertions, deletions or substitutions) that will convert the shorter sequence to that region. Thus, the alignment scoring scheme is designed to model sequencing errors, rather than evolutionary processes. The program can align a 100 kb sequence to a 1 megabase sequence in a few seconds on a workstation, provided that there are very few differences between the shorter sequence and some region in the longer sequence. The program has been used to assemble sequence data for the Genomes Division at the National Center for Biotechnology Information.*

Availability: *A version of sim3 for UNIX machines can be obtained by anonymous ftp from ncbi. nlm. nih. gov, in the pub/sim3 directory.*

Contact: *For portable versions for Macs and PCs, contact zjing@sunset. nlm. nih. gov.*

Introduction

*Sim*3 computes (or attempts to compute, under a heuristic option available to the user) an alignment that minimizes the sum of the following costs: (i) except for the end gaps of the shorter sequence (which are not penalized), a gap of length k in either sequence is penalized at the cost of $k+1$; (ii) each mismatch costs 1. Adding 1 to a gap's length to derive its cost decreases the likelihood of generating gaps that are separated by only a few paired nucleotides. Figure 1 illustrates the scoring scheme.

Our algorithm runs in two phases. Phase 1 locates the interval in the longer sequence that should be aligned with the shorter sequence. Phase 2 employs a divide-and-conquer approach to determine that alignment. The end gaps, if they exist, are then added to the alignment. Figure 2 illustrates the two-phase approach, using the familiar representation of a pairwise alignment as path through the dynamic programming grid.

Two methods are provided for accomplishing Phase 1. The first, called 'heuristic Phase 1', runs much faster than the second, but only the second can be guaranteed to produce an optimal subinterval of the longer sequence. Phase 2, whose computation time is insignificant if the alignment cost is small, is always solved exactly. In addition to choosing which approach is to be used for Phase 1, the user can provide an upper bound on the alignment cost; this prevents the program from computing aimlessly if a pair of unrelated sequences is inadvertently given.

*Sim*3 was written in C using the NCBI software toolkit. The input and output formats are the same as the previously published *sim*2 algorithm (Chao *et al.*, 1994). *Sim*3 is able to align one target sequence to many related sequences, and can display the results as one-to-many multiple pairwise alignments. Also, it can take input sequences either locally, as FASTA formatted files or structured ASN. 1 files, or remotely, by retrieving them from Network Entrez (Schuler *et al.*, 1995) given the locus names or accession numbers. The results are stored as ASN. 1 Seq-align objects, which can be viewed textually or exported to such graphic browsers as ChromoScope (Zhang *et al.*, 1994) or the new NCBI sequence submission tool, Sequin (Kans, 1996). *Sim*3 has been run successfully on UNIX machines, on the Macintosh, and under Microsoft Windows 95 and Windows NT on the PC.

Algorithm

Let $A = a_0a_1a_2 \dots a_{M-1}$ and $B = b_0\, b_1\, b_2 \dots b_{N-1}$ be two sequences of lengths M and N, respectively, where without loss of generality $N \geq M$. The edit graph for sequences A and B is a directed graph with a vertex at each integer grid point (x, y), $0 \leq x \leq M$ and $0 \leq y \leq N$.

Let $I(k, c)$ denote the x value of the farthest point in diagonal k that can be reached from the source [i. e. grid point (0, 0)] with cost c and that is *free* to open an insertion gap. That is, the grid point can be (i) reached by a path of cost c that ends with an insertion, or (ii) reached by any path of cost $c-1$ and the gap-open penalty of 1 can be 'paid in advance'.

Department of Computer Science and Information Management, Providence University, Shalu, Taichung, Taiwan 43309, [1]National Center for Biotechnology Information, National Library of Medicine, NIH, Bethesda, MD 20894 and [2]Department of Computer Science and Engineering, The Pennsylvania State University, University Park, PA 16802, USA

[3]*To whom correspondence should be addressed*

[The more traditional definition, which considers only case (i), results in the storage of more vectors.] Let $D(k, c)$ denote the x value of the farthest point in diagonal k that can be reached from the source with cost c and is *free* to open a deletion gap. Let $S(k, c)$ denote the x value of the farthest point in diagonal k that can be reached from the source with cost c. With proper initializations, these vectors can be computed by the following recurrence relation:

$$I(k,c) = \max\{I(k-1,c-1), S(k,c-1)\}$$

$$D(k,c) = \max\{D(k+1,c-1)+1, S(k,c-1)\}$$

$$S(k,c) = snake(k, \max\{S(k,c-1)+1, I(k,c), D(k,c)\})$$

$$\text{where } snake(k,x) = \max\{x, \max\{z : a_x \ldots a_{z-1} = b_{x+k} \ldots b_{z-1+k}\}\}.$$

Since vectors at cost c depend only on those at costs c and $c-1$, it is straightforward to derive a linear-space version of the above recurrence relationships.

The heuristic Phase 1

Let f_i denote the number of w-mers starting between positions i and $i+nw-1$ of the longer sequence that are also a w-mer starting between positions 0 and $nw-1$ of the shorter sequences (see Figure 3). (In other words, we count those w-mers that can be in both the first nw w-mers from position i of the longer sequences and the first nw w-mers of the shorter sequence.) It should be noted that we do not take into account the order of w-mers. Similarly, let b_i denote the number of w-mers ending between positions $i-nw+1$ and i of the longer sequence that are also a w-mer ending between positions $M-nw$ and $M-1$ of the shorter sequences.

Let *maxf* (*maxb*) be the maximum value of all such f_i (b_i) values. We want to determine an ordered list *FRONT* containing all those positions i such that $f_i = maxf$ and an ordered list *BACK* containing all those positions i such that $b_i = maxb$. This is done by first building a hash table with 4^w entries, where the entry value can be 0, 1, 2, 3. The entry value is 3 if its corresponding w-mer occurs inthe first and the last nw w-mers of the shorter sequence; it is 2 if its corresponding w-mer occurs only in the last nw w-mers; it is 1 if its corresponding w-mer occurs only in the first nw w-mers; it is 0 otherwise. The next step is to compute f_i and b_i. Initially, we compute f_0 and b_{nw+w-2}. With two shift operations, we can compute f_1 and b_{nw+w-1}. This procedure is repeated until we have reached the end of the longer sequence, updating the lists *FRONT* and *BACK* as we go. If *maxf* (or *maxb*) falls below a certain percentage of nw, we suggest that an exact Phase 1 should be used. Finally, we select a pair of positions in *FRONT* and *BACK* such that the distance between them is closest to the length of the shorter sequence.

CCATGCAAATGCCAT
--ATGCA--TCCCA-

Fig. 1. The total cost in this example is 4 since a mismatch costs 1 and an internal gap of length 2 costs 3.

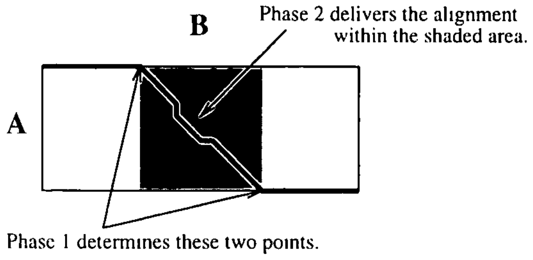
B
Phase 2 delivers the alignment within the shaded area.
A
Phase 1 determines these two points.

Fig. 2. Two-phase approach to aligning a shorter sequence A with a longer sequence B. The thicker line denotes the alignment.

The exact Phase 1

Phase 1 can be accomplished by applying the recurrences for I, D and S, where all costs in row 0 are initialized to 0. Once row M is reached, the desired interval has been located. Although the worst-case running time for this approach is $O(MN)$, the average running time is $O(N \times Dist)$, where *Dist* is the distance of A and B. This holds because the average length of a *snaked* fragment is a small constant. For random DNA sequences, the average length is $\frac{4}{9} = \sum_{i=1}^{\infty} \frac{i}{4^i}$.

Phase 2

To deliver the alignment, we divide the problem at the middle cost, and recursively solve each subproblem, much as was done by Myers (1986) and Miller and Myers (1988). To do so, we need the following 'backward' vectors. Let $\bar{I}(k, c)$ denote the x value of the farthest I node in diagonal k that can reach the sink [i. e. grid point (M, N)] with cost c. Let $\bar{D}(k, c)$ denote the x value of the farthest D node in diagonal k that can reach the sink with cost c. Let $\bar{S}(k, c)$ denote the x value of the farthest S node in diagonal k that can reach the sink with cost c. With proper initializations, these vectors can be computed by the following recurrence relation:

$$\bar{S}(k,c) = \overline{snake}(k, \min\{\bar{S}(k,c-1)-1, \bar{D}(k,c-1), \bar{I}(k,c-1)\})$$

$$\bar{D}(k,c) = \min\{\bar{D}(k-1,c-1)-1, \bar{S}(k,c)\}$$

$$\bar{I}(k,c) = \min\{\bar{I}(k+1,c-1), \bar{S}(k,c)\}$$

$$\text{where } \overline{snake}(k,x) = \min\{x, \min\{z : a_z \ldots a_{x-1} = b_{z+k} \ldots b_{x-1+k}\}\}.$$

As before, it is straightforward to derive a linear-space version of the recurrence relation.

Figure 4 gives the pseudo-code for delivering the alignment in linear space. Line 12 determines the correct partition

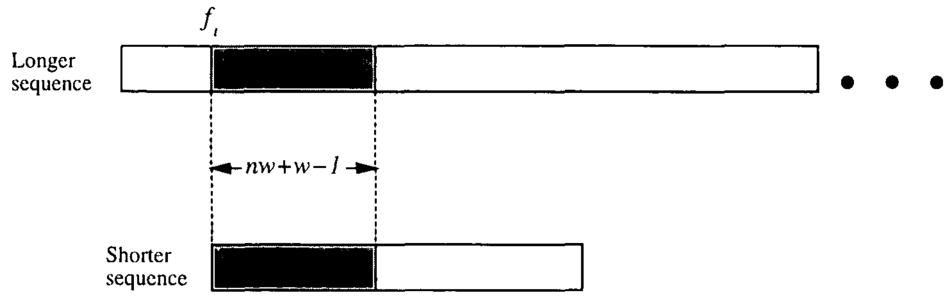

Fig. 3. An illustration of f_i.

point because, for any specific diagonal, vectors S, D and I are monotonically increasing with respect to the cost, while $\bar{S}$, $\bar{D}$ and $\bar{I}$ are monotonically decreasing. The time complexity of Phase 2 is $O(N \times Dist)$.

Implementation

The heuristic first phase uses array storage for 4^w characters to represent the hash table. Its time complexity is $O(M+N)$ plus $O(4^w)$ time for building the hash table. *FRONT* and *BACK* are each of size at most M. In contrast, the exact first phase uses array storage for $6M+6N+12$ integers. The second phase recycles all the storage from the first phase and allocates array storage for $12D+6$ integers, where D is the distance between the sequences.

Example

As molecular biology enters the era of large-scale sequencing, an unprecedented volume of relatively unannotated sequence data, which includes tens to hundreds of megabases of human genomic sequences, is expected to be submitted to the public databases each year. It is almost impossible to study the biological functions of those large genomic sequences if they are not integrated with the other related biological data, such as the cDNA/mRNA sequences, the EST sequences, and the STS markers mapped by various methods. Sequence alignment is the most common and essential computational analysis to establish the relationships within this vast collection of data. *Sim*3 is an indispensable tool for genome research because it can compute alignments for long DNA sequences and is fast enough to handle the large data volume. Recently, the National Center for Biotechnology Information (NCBI) created a Genomes Division (J.Zhang and J.Ostell, unpublished) for the data retrieval system Entrez (Schuler *et al.*, 1996), which organizes sequence data for model organisms and provides an integrated graphic view of the genetic, physical, cytogenetic and sequence data. *Sim*3 has been used extensively to compile the non-redundant sequence maps for model organisms in the Genomes Division.

It is a daunting task to organize sequence data for a model organism. The problem is complicated by the huge volume and rapid growth of the sequence data, the high level of data redundancy, and the hierarchical relationships between the genomic sequences, the cDNA/mRNA sequences, and the EST and STS sequences complicate the problem. The most complex case is the human genome. As of June 1996, there were a total of 217 774 765 bp of 497 953 human sequences with ~7-fold redundancy (J.Zhang, unpublished) in the public sequence databases. On average, <1% of the genome is sequenced. Each chromosome consists of islands of known genomic sequences separated by large, unsequenced gaps. One genomic sequence may encode multiple genes. One gene may have several alternatively spliced transcript sequences,

```
procedure $path(I_1, J_1, Type_1, I_2, J_2, Type_2, Dist)$
  { if boundary cases then
      { Output the edit script; return; }
    else
      {
         $mid \leftarrow Dist/2$
         $\overline{mid} \leftarrow Dist - mid$
         A linear-space forward pass computes $S(k, mid)$, $D(k, mid)$, and $I(k, mid)$
             for $J_1 - I_1 - mid \le k \le J_1 - I_1 + mid$.
         A linear-space backward pass computes $\bar{S}(k, \overline{mid})$, $\bar{D}(k, \overline{mid})$, and $\bar{I}(k, \overline{mid})$
             for $J_2 - I_2 - \overline{mid} \le k \le J_2 - I_2 + \overline{mid}$.
         Let $K$ be the diagonal such that $X(K, mid) \ge \bar{X}(K, \overline{mid})$, where $X$ is $S$, $D$, or $I$.
         $path(I_1, J_1, Type_1, X(K, mid), X(K, mid) + K, X, mid)$
         $path(X(K, mid), X(K, mid) + K, X, I_2, J_2, Type_2, \overline{mid})$
      }
  }
```

Fig. 4. Delivering the alignment in linear space.

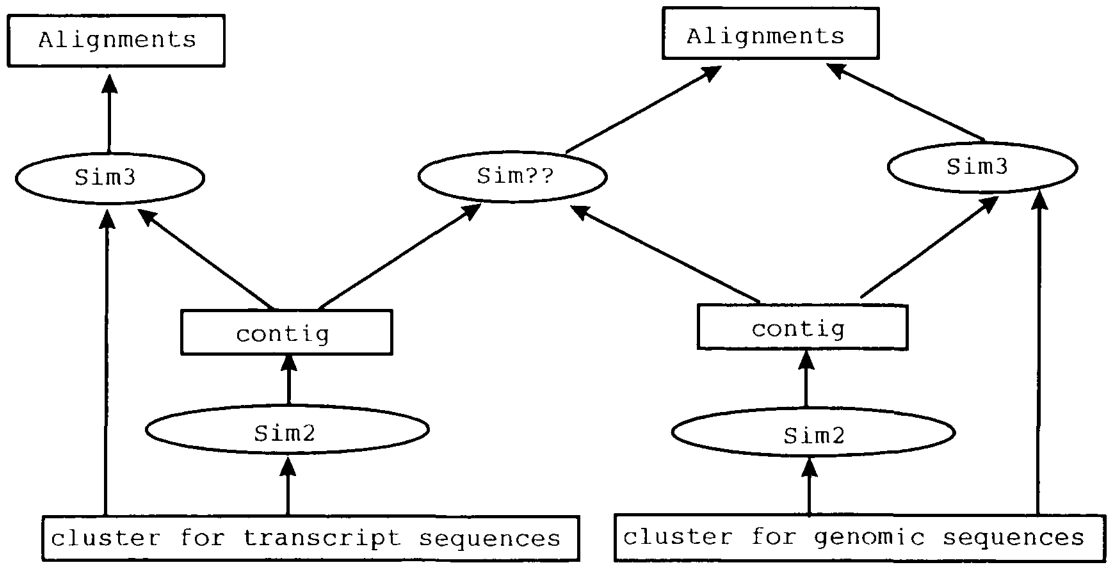
Alignments
Alignments
Sim3
Sim??
Sim3
contig
contig
Sim2
Sim2
cluster for transcript sequences
cluster for genomic sequences

Fig. 5. Construction of the non-redundant sequence maps for the human genome. The ovals show the algorithms and the rectangles are the input and output data at each stage. The arrows indicate the directions of the data flow. Local alignments were computed with *sim2* to build the contig sequence for a cluster of transcript or genomic sequences. Each member of the cluster is aligned to the contig with *sim3*. The contig for a transcript sequence is also aligned to its parental cluster for the genomic sequences to represent the hierarchical relationship. A more rigorous algorithm is in development to compute the alignment between the transcript sequence and the genomic sequence.

and one transcript sequence may have fragments that were sequenced multiple times to give ESTs. In the Genomes Division, a non-redundant sequence map was compiled for each chromosome, and the complex relationships among the sequence data are represented by optimal, gapped alignments computed with the *sim* (Huang *et al.*, 1990), *sim2* and *sim3* programs. Both *sim* and *sim2* compute the top *K* (arbitrary) non-intersecting local alignments for very long sequences, while *sim3* computes a global alignment.

Figure 5 illustrates the various stages of the process. We use the ADH2 gene on human chromosome 4 as an example. Sequences at the same level of genome organization were treated as a cluster. For the ADH2 gene, transcript sequences were grouped as a UniGene (Boguski and Schuler, 1995) cluster. Local alignments were computed with the *sim2* program to build a contig from the overlapping sequences. The contig served as the master sequence and each member was aligned to it with *sim3*. Figure 6 is the graphic view of the alignments between the master sequence and the ADH2 transcript sequences. There are four mRNA and three EST sequences in this cluster. The master sequence gives the non-redundant view of the data, while the alignments

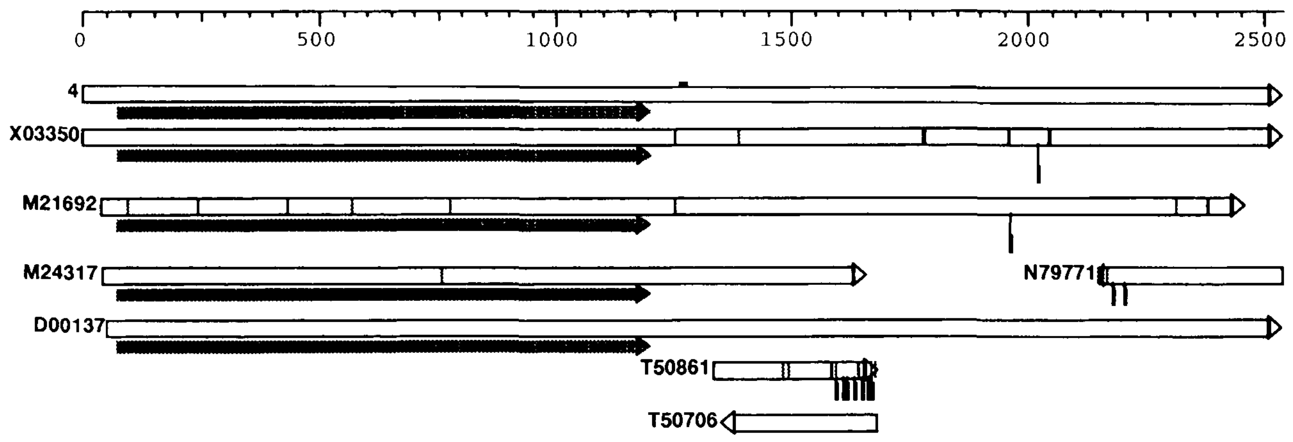

Fig. 6. A graphical view of the alignments of the transcript sequences for the ADH2 gene. The master sequence is assigned the UniGene cluster id 4. There are four mRNAs (accessions X03350, M21692, M24317 and D00137) and three EST sequences (accessions T50861, T50706 and N79771) in the cluster. The dark lines underneath the contig and the mRNA sequences mark the annotated coding region features. A vertical line attached with a box indicates an insertion in the aligned sequences. There are insertions at the 3′ UTR of X03350 and M21692. Mismatches are marked by vertical bars inside the sequence box. A single line connecting two adjacent sequence boxes represents a gap (see Figure 8 for the gaps between the genomic sequence and the transcript sequence)

```
                         10        20        30        40        50
                    |    |    |    |    |    |    |    |    |    |
4           > 1178  tccgtaccgtcctgacgttttgaggcaatagagatgccttcccctgtagc
alcohol de  > 369   I  R  T  V  L  T  F ^^^
X03350      > 1178  . ..................... .. .......................
ADH beta-1  > 369   I  R  T  V  L  T  F ^^^
M21692      > 1138  ...... ...........................................
alcohol de  > 369   I  R  T  V  L  T  F ^^^
M24317      > 1136  .......... ... .............. ....................
alcohol de  > 369   I  R  T  V  L  T  F ^^^
D00137      > 1126  .......... .......................................
alcohol de  > 369   I  R  T  V  L  T  F ^^^

                         10        20        30        40        50
                    |    |    |    |    |    |    |    |    |    |
4           > 1228  agtcttcagcctcctctaccctacaagatctggagcaacagct
X03350      > 1228  .............    .......g... ..............
M21692      > 1188  .... .............. . ..g.........  ........
M24317      > 1186  .........................................
D00137      > 1176  . . ....................................  ......
```

Fig. 7. Text view of the alignments of the transcript sequences for the ADH2 gene. The first line is the DNA sequence, the second is the protein sequence from the coding region. The stop codon is marked with the triplet '^^^' symbol. If the aligned nucleotide is identical to the master, it is indicated as a dot; otherwise the actual base is presented. The boxed region may suggest a potential polymorphism.

show sequence variations. Some of the variations can be attributed to sequencing errors. There are increased sequence discrepancies at the 3′ end of the two EST sequences (accessions N79771 and T50861), which is consistent with the decreased resolution at the end of the sequencing gel. Other discrepancies may suggest potential polymorphism. In the 3′ untranslated region, at position 1253 of the master sequence (the boxed region in Figure 7), two mRNAs (accessions X03350 and M21692) have residue '*g*' and the other two mRNAs (accessions M24317 and D00137) have residue '*a*'. One mRNA sequence (accession M24317) has a shorter 3′ untranslated region than the others because of alternative splicing.

The same procedure was also applied to the genomic sequences. Figure 8 shows the alignments for the genomic cluster of ADH2 sequences, which includes two genomic sequences (accessions SEG_HUMADH2S0 and SEG_HUMADH2E) and one STS sequence (accession G05714). The genomic contig defines an HGM (Human GenoMic) cluster in the Genomes Division. The speed of *sim*3 enables it to complete all the alignments for the entire human genome overnight. The hierarchical relationship between the transcript cluster and its parental genomic cluster is also represented by a sequence alignment. In Figure 8, the UniGene contig is aligned to the HGM cluster of the ADH2 gene. The alignment reflects the coding-region structure of the genomic sequence, with aligned regions corresponding to exons and gaps covering the introns. Currently, we do not have a rigorous algorithm to align a genomic sequence to its transcript sequence if there are large introns in the coding region. For now, we use *sim* to compute the alignment with the gap extension penalty set to zero. A variation of *sim*3 is in development which will be able to handle large gap size. The HGM clusters will then be localized on the chromosome by a number of methods (J.Zhang and J.Ostell, in preparation). Entrez Genomes

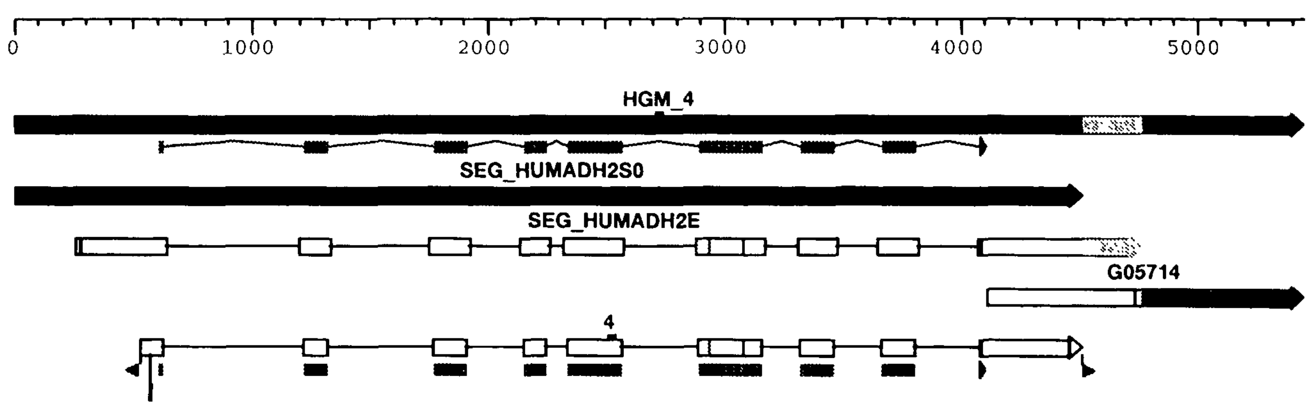

Fig. 8. A graphical view of the alignments of the genomic sequences for the ADH2 gene. The master sequence (HGM_4) was constructed by merging the three overlapping sequences, SEG_HUMADH2S0, SEG_HUMADH2E and G05714. The coding region features are displayed together with the alignments.

Division can be browsed either via the World Wide Web (http://www.ncbi.nlm.nih.gov) or with the network Entrez software, which can be downloaded from the NCBI anonymous ftp site (ncbi.nlm.nih.gov).

Discussion

*Sim*3 is one of a suite of pairwise alignment tools that we have developed. Each of them is designed to deal with genomic DNA sequences, where a primary concern is with the potentially huge length of the alignment. Three of the programs, called *blast*, *sim* and *sim*2, aim to find all of the significant matching regions, allowing for repeated regions (such as a family of related genes) within each sequence. Blast (Schwartz *et al.*, 1991) is a much simplified version of a database searching program of the same name (Altschul *et al.*, 1990), and reports only gap-free alignments. *Sim* permits gaps, and hence can find matches overlooked by *blast*, but typically runs a few thousand times slower (e.g. 10 h as opposed to 10 s for sequences of length 100 000). *Sim*2 (Chao *et al.*, 1995) attempts to attain *sim*'s sensitivity with improved time efficiency.

*Sim*3 provides a very efficient solution for a much more specialized problem. It computes just a single continuous alignment that is required to include all of the shorter sequence. In particular, *sim*3 is not intended for comparison of a cDNA sequence with genomic DNA, allowing for introns. However, in its favor, when the shorter sequence is very similar to a region of the longer sequence, *sim*3 runs faster than even *blast*, and for certain applications its output is much more useful. For instance, on our Sparc 5 workstation, *sim*3 takes 1.5 s to compare the 91 414 nucleotides of GenBank Locus ECOUW85 with a 726 039 nucleotide contig containing it (modulo 23 nucleotides inserted, deleted, or replaced), whereas *blast* takes 30 s. Moreover, compared with *sim* and especially *sim*2, it is extremely simple to use; one need not be concerned with correct alignment-scoring parameters.

The algorithmic underpinnings of *sim*3 include 'greedy' algorithms for solving a simple formulation of the alignment problem called the 'longest common subsequence problem', which is equivalent to finding the fewest (one-character) insertion and deletion operations that will convert one given sequence to another. Let M and N be the two sequence lengths, and let E denote that minimum number of operations (i.e. the *edit distance* between the two sequences). Greedy alignment algorithms (Ukkonen, 1985; Myers, 1986; Wu *et al.*, 1990) are ideally suited to the case where E is very small compared to M and N; they run in worst-case time $O(\min(M,N)\cdot E)$ and space $O(M+N)$ [Myers (1986) first obtained that space bound with a greedy approach]. Greedy alignment algorithms have been implemented for determining the lines that need to be inserted or deleted to convert one given text file to another (Miller and Myers, 1985; Miller, 1987) and can be generalized to produce optimal sets of editing operations when substitutions are counted and when a run of k consecutive inserted (or deletions) letters is given the score $\alpha + \beta k$ for fixed α and β (Myers and Miller, 1989). Greedy algorithms can also accommodate symbol-dependent substitution costs (e.g. a penalty for transversions that differs from the penalty for transitions), but we have not chosen to implement that option. Other algorithms (Kumar and Rangan, 1987; Apostolico *et al.*, 1992) have been developed to solve the longest common subsequence problem in these same worst-case time and space bounds, but greedy algorithms have the advantage of a much better *expected-case* time bound of $O(NE^2)$.

Acknowledgements

W.M. was supported by grant R01 LM05110 from the National Library of Medicine. K.-M.Chao was supported by grant NSC 86-2213-E-126-002 from The National Science Council, Taiwan.

References

Altschul,S., Gish,W., Miller,W., Myers,E. and Lipman,D. (1990) A basic local alignment search tool. *J. Mol. Biol.*, **215**, 403–410.
Apostolico,A., Browne,S. and Guerra,C. (1992) Fast linear-space computations of longest common subsequences. *Theor. Comput. Sci.*, **92**, 3–17.
Boguski,M.S. and Schuler,G.D. (1995) Establishing a human transcript map. *Nature Genet.*, **10**, 369–371.
Chao,K.-M., Zhang,J., Ostell,J. and Miller,W. (1995) A local alignment tool for very long DNA sequences. *Comput. Applic. Biosci.*, **11**, 147–153.
Huang,X., Hardison,R. and Miller,W. (1990) A space-efficient algorithm for local similarities. *Comput. Applic. Biosci.*, **6**, 373–381.
Kans,J.A. (1996) *Genome Mapping & Sequencing*. Sequin. pp. 114.
Kumar,X. and Rangan,C. (1987) A linear-space algorithm for the LCS problem. *Acta Inf.*, **24**, 353–362.
Miller,W. (1987) *A Software Tools Sampler.* Prentice-Hall.
Miller,W. and Myers,E. (1985) A file comparison program. *Software—Practice Experience*, **15**, 1025–1040.
Miller,W. and Myers,E. (1988) A simple row-replacement method. *Software—Practice Experience*, **18**, 597–611.
Myers,E. (1986) An *O(ND)* difference algorithm and its variations. *Algorithmica*, **1**, 251–266.
Myers,E. and Miller,W. (1989) Row replacement algorithms for screen editors. *ACM Trans. Prog. Lang. Syst.*, **11**, 33–56.
Schuler,G.D., Epstein,J.A., Ohkawa,H. and Kans,J.A. (1996) Entrez: molecular biology database and retrieval system. *Methods Enzymol.*, **266**, 141–162.
Schwartz,S., Miller,W., Yang,C.-M. and Hardison,R. (1991) Software tools for analyzing pairwise sequence alignments. *Nucleic Acids Res.*, **19**, 4663–4667.
Ukkonen,E. (1985) Algorithms for approximate string matching. *Inf. Control*, **64**, 100–118.
Wu,S., Manber,U., Myers,E. and Miller,W. (1990) An *O(NP)* sequence comparison algorithm. *Inf. Process. Lett.*, **35**, 317–323.
Zhang,J., Ostell,J. and Rudd,K. (1994) ChromoScope: A graphic interactive browser for *E. coli* data expressed in the NCBI data model. *Proceedings of the Twenty-Seventh Annual Hawaii International Conference on System Sciences V*, pp. 58–67.

Received on August 14, 1996; accepted on October 8, 1996