Open access • Journal Article • DOI:10.1093/BIOINFORMATICS/11.2.147

A local alignment tool for very long DNA sequences. — Source link

Kun-Mao Chao, Jinghui Zhang, James Ostell, Webb Miller

Institutions: Pennsylvania State University, National Institutes of Health

Published on: 01 Apr 1995 - Bioinformatics (Oxford University Press)

Topics: Smith–Waterman algorithm

A Local Alignment Tool for Very Long DNA Sequences

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

Abstract. This paper presents a practical program, called *sim2*, for building local alignments of two sequences, each of which may be hundreds of kilobases long. *Sim2* first constructs n best non-intersecting chains of ''fragments,'' such as all occurrences of identical 5-tuples in each of two DNA sequences, for any specified $n \geq 1$. Each chain is then refined by delivering an optimal alignment in a region delimited by the chain. *Sim2* requires only space proportional to the size of the input sequences and the output alignments, and the same source code runs on UNIX machines, on Macintosh, on PC, and on DEC ALPHA PC. We also describe an application of *sim2* for aligning long DNA sequences from *E. coli*. *Sim2* facilitates contig-building by providing a complete view of the related sequences, so differences can be analyzed and inconsistencies resolved. Examples are shown using the alignment display and editing functions from the software tool, *ChromoScope*.

Key Words. Sequence comparison, Local alignment, Dynamic programming, Constrained alignment, Linear-space algorithm.

1. Introduction.

The *local alignment* problem (for two sequences) is to find an alignment that is highest-scoring among all alignments between an arbitrary section of the first sequence and an arbitrary section of the second sequence (Smith and Waterman, 1981). Probably the most useful variant for aligning biological sequences is that of computing ''n best non-intersecting'' local alignments, for which Waterman and Eggert (1987) gave a very efficient dynamic programming algorithm. A version of their approach using only space proportional to the sum of the sequence lengths was developed by Huang *et al* (1990), and implemented in a program called *sim*. Though these methods are more sensitive than the approach described here, they are much slower in general.

To attain greater speed, Wilbur and Lipman (1983, 1984) employed the strategy of building alignments from fragments. For example, one could specify some fragment length $k \geq 1$ and work with alignment-fragments consisting of exact matches of length at least k. With protein sequences, it might well work better to begin with inexact but high-scoring matches, such as those used by the *blast* program (Altschul *et al.*, 1990) for other purposes. In any case, algorithms that optimize the score over alignments constructed from fragments can run faster than algorithms that optimize over all possible alignments. Moreover, a full-resolution alignment process can be made more efficient by restricting the search to a ''neighborhood'' of the chain of fragments (Pearson and Lipman, 1988).

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

To balance speed and sensitivity, we have developed a new alignment tool, *sim2*, that can efficiently detect alignments containing gaps. Moreover, *sim2* requires only space proportional to the size of the input sequences and the output alignments. *Sim2* runs in two phases. The first phase generates n best chains of fragments, while the second refines each chain by applying a more traditional dynamic-programming algorithm in a region delimited by the chain. *Sim2* is designed so that these two parts can themselves also be stand-alone tools or subroutines.

A few variants of *sim2* are also provided. For example, *sim2* can be used to compute n best local alignments between two DNA sequences in both orientations. It can also select only alignments with scores no less than a cut-off score, or compute alignments within a user-selected region from each sequence. A modified version, *sim2aln*, is designed for one-to-many sequence alignment, which can be used as a post-process for *blast* search.

Sim2 is written in C using the NCBI software toolkit. It currently runs on UNIX machines, on the Macintosh, under Microsoft Window on PC, and under Window NT on ALPHA PC. It can be obtained by anonymous ftp; contact webb@cse.psu.edu for instructions.

2. Methods.

Basically, *sim2* implements two alignment techniques whose theoretical bases have been described in earlier papers. The first algorithm constructs a rather crude alignment by efficiently linking together very short alignments, using a method described by Chao and Miller (1994). The initially approximation is then refined down to the level of individual matching nucleotides using a technique developed by Chao *et al.* (1993). This section outlines the two methods in somewhat more detail.

2.1. Building n best non-intersecting chains of fragments.

The ‘‘very short alignments’’ mentioned above are defined as follows. Let the given sequences be $A = a_1 a_2 \cdots a_M$ and $B = b_1 b_2 \cdots b_N$. Define a *fragment* to be a triple (i, j, k) such that $a_i = b_j$, $a_{i+1} = b_{j+1}$, $\cdots$, $a_{i+k-1} = b_{j+k-1}$ and $k \geq k\_min$, for fixed minimum fragment length k_min. Moreover, a fragment is to be maximal, i.e., not properly contained in another fragment. Fragment $f' = (i', j', k')$ is said to be *above* fragment $f = (i, j, k)$ if $i' + k' \leq i$ and $j' + k' \leq j$. Notice that this defines a partial-ordering relation. A chain of fragments is defined as a sequence of fragments $f_1, f_2, \cdots, f_l$ such that f_i is above f_{i+1} for $i = 1, \cdots, l-1$.

We also need an objective criterion for preferring one fragment chain over another. The score of fragment $f = (i, j, k)$ is defined as k and denoted $sc(f)$. Penalties for connecting fragments are needed for scoring chains. Let the nonnegative constant g and positive constants e and r be the penalties for opening up a gap (horizontal or vertical displacement), for extending a gap by one symbol, and for replacing one symbol by another, respectively. The affine function $gap(t) = g + te$ is charged for a gap of length t. Define the *diagonal* of fragment $f = (i, j, k)$ to be $j - i$, denoted by $Diag(f)$. Suppose $f' = (i', j', k')$ is above $f = (i, j, k)$. The penalty of connecting f' and f, denoted by $Connect(f', f)$, is defined as follows:

Case 1: $Diag(f) = Diag(f')$:
$$Connect(f',f) = (i - i' - k')r.$$
Case 2: $Diag(f) > Diag(f')$:
$$Connect(f',f) = gap(Diag(f) - Diag(f')) + (i - i' - k')r.$$
Case 3: $Diag(f) < Diag(f')$:
$$Connect(f',f) = gap(Diag(f') - Diag(f)) + (j - j' - k')r.$$

The *score* of a chain $(f_1, f_2, \cdots, f_l)$ is defined as the sum of the fragment scores, minus the connection penalties of adjacent fragments, i.e.:

$$\sum_{i=1}^{l} sc(f_i) - \sum_{i=1}^{l-1} Connect(f_i, f_{i+1})$$

Chao and Miller (1994) combine a time-efficient algorithm developed by Eppstein *et al.* (1992) with a space-saving approach of Hirschberg (1975) to obtain an algorithm that uses $O((M+N+F\log N)\log M)$ time and $O(M+N)$ space to deliver the best chain of fragments from a pool of F fragments. They then give a time-and-space efficient algorithm to compute n best non-intersecting chains of fragments following the general outline of Huang and Miller (1991). (Here, ‘‘non-intersecting’’ means that no fragment appears in more than one chain.)

2.2. Refining the chains

To refine the chains of fragments, *sim2* delivers an optimal alignment in the neighborhood of each chain. For region bounds, *sim2* uses the upper and lower envelopes of all upright rectangles extending from the end of the ith fragment to the start of the $(i+2)$th, for relevant i, as pictured in Figure 1. It should be noted that other ways of defining region bounds sometimes work better. For example, *sim2* supports the alternative of determining the region by defining a band of fixed width centered on each fragment and connecting the bands with rectangles.

Chao *et al.* (1993) give an algorithm that requires $O(R)$ time and $O(M+N)$ space to align two sequences when the alignment is constrained to lie between two arbitrary boundary lines in the dynamic programming matrix, where R is the area of the feasible region. To attain this efficient use of both space and time, the paper develops a strategy for problem division that differs substantially from the classic approach of Hirschberg.

The following minor alteration to the previously published strategy proved to be useful. In the first phase of *sim2*, two maximal fragments can be chained together only if the end position of one fragment is topologically smaller than the starting position of the other. This occasionally keeps *sim2* from utilizing particular fragments, resulting in an inferior alignment score. To avoid this problem, the definition of *above* in Section 2 can be altered to: fragment $f'=(i',j',k')$ is said to be *above* fragment $f=(i,j,k)$ if $i' < i$ and $j' < j$. The penalty of connecting f' and f can be redefined accordingly. However, it remains open whether the algorithms used in Chao and Miller (1994) can be adapted for this modification. *Sim2* employs an *ad hoc* remedy for this problem. During the second phase, both ends of the fragment chain are extended outward using dynamic-programming. The rationale for giving special treatment to the ends is that any useful partial fragment lying near the internal part of the fragment chain is automatically considered in the second phase of *sim2*. This strategy proved successful in the experiments we conducted.

3. Implementation.

Space requirements

The first phase of *sim2* uses array storage for $7M + 8N$ integers and $2M + 2N$ characters. Candidate lists are implemented as skip lists (Pugh, 1990). At most $3M + 3N$ elements will be in the lists, where each element uses three integers and, on average, $4/3$ pointers, using $p = 0.25$ (Pugh, 1990). Therefore, the candidate lists require $13M + 13N$ integers. In total, the first phase requires storage for $20M + 21N$ integers and $2M + 2N$ characters. Since fragments are found using hashing (Altschul *et al.*, 1990), additional storage for hash tables is needed. The second phase of *sim2* uses only array storage for about $4M + 8N$ integers and $3M + 5N$ characters (Chao *et al.*, 1993), which is dominated by the space for the first phase.

I/O

Sim2 can take either a *FASTA*-formatted text file or a structured ASN.1 Seq-entry object. Alternatively, it can retrieve sequences on-line from *Entrez* given the locus name or the accession number. All the different forms are converted into *Sequence location objects*, which specify the start, the stop and the orientation on each sequence. The output is stored as a ASN.1 *Seq-align* object, which can either be exported as a file or attached to the sequence object as the history for the sequence.

4. Example.

Since *sim2* aligns long sequences quickly, it is a valuable analytic tool for genome-sized sequencing projects. It can detect internal sequence inconsistencies, reveal polymorphism in certain regions, and check for sequencing errors in the published record. As an experiment, we applied *sim2* to the production of the integrated *E. coli* genome map, EcoSeq7 (K. Rudd, personal communication).

E. coli is a very well studied organism, with currently 60% of its 4.67×10^6 bp genome sequenced. While some research groups intend to sequence the whole genome and have produced several sequences over 100,000 bp long (Sofia *et al*., 1994; Burland *et al*., 1993; Daniels *et al*., 1992; Plunkett *et al*., 1993; Blattner *et al*., 1993), most sequence data have been generated as independent, non-coordinated efforts. Rudd (1993) has compiled a non-redundant sequence map by merging overlapping sequences. However, this process does not allow for display of the differences in all the published sequence records. To provide a complete view of all the published *E. coli* sequences while maintaining the consensus compiled in EcoSeq7, we need to align all available *E. coli* sequence data to each contig to provide a sequence history. First we do a *blast* search of each contig in EcoSeq7 to generate a list of *E. coli* sequences that match the contig. Then we apply *sim2* to compute a pairwise alignment between the contig and each of its *blast* hits. The longest contig we have encountered, ggt-ecoM, is 726,039 bp long. It had 380 *blast* hits, three of which are more than 100,000 bp. *Sim2* was able to complete the 380 alignments overnight. It took about four hours to finish aligning the rest of the contigs in the *E. coli* genome.

Our experience proves that it is feasible to use *sim2* to produce daily updates of sequence history for a genome as large as *E. coli*.

The sequence history of each contig is stored as a one-to-many alignment, which can be displayed graphically in the alignment viewer of *ChromoScope* (Zhang *et al.*, 1994), showing gaps, insertions and mismatches (Figure 2). The graphic output can help a curator to check internal consistency, so that the alignments agree with the instructions used to build the contigs. For example, in Figure 2, there is an insertion and a deletion in constituent ECOUW82 and there is a gap in constituent yigUeco. A curator can easily find those trouble spots and resolve the discrepancies by rebuilding the contig with the correct instructions. As an effort to automate the process of contig-building, we have incorporated *sim2* as part of the genome editor in *ChromoScope*, so consistency can be checked immediately after a contig is built.

A large sequence generated from a genome sequencing group may cover regions that were previously sequenced by other single-gene research groups, and those sequences may have conflicts with each other. Aligning those sequences with *sim2* is a good way to visualize all sides of the data. As an example, we aligned one of the large sequences generated from the *E. coli* genome sequencing group in University of Wisconsin (Burland *et al.*, 1993), ECOUW82 (Genbank Accession L10328, length 136254), with 66 related sequences in GenBank. The alignment is shown in Figure 3. Sometimes, sequence discrepancies are caused by polymorphism, which can be observed in the coding region for protein tryptophanase (Figure 4). Two sequences aligned to this region, ECOTNAA (Accession K00032) and ECLTIL (Accession X15974). Compared with ECOTNAA, ECLTIL has a lower nucleotide sequence identity to ECOUW82. However, the protein sequences are identical. Since both ECOTNAA and ECOUW82 are sequences for *E. coli K12* while ECLTIL is a sequence for *E. coli B/1t7-A*, DNA sequence polymorphism in a different strain for the same gene is the probable explanation. In some cases, each sequencing group has its own view about the ambiguities for the published data. For example, in the coding region for DNA recombinase, at positions 15951, 15952 in ECOUW82, the researchers are uncertain if a G or a C should be presented at those two positions. Two other groups sequenced the same gene and have a different point of view: both Genbank entry ECORECG (Accession M64367) and EMBL entry ECRCG (Accession X59550) have a GC pair in their alignment (Figure 6). Sometimes, differences come from the same group. As shown in Figure 5, sequence A and sequence B aligned to the same region on ECOUW82, but B has a frame-shift in one of its coding regions, while A does not have the problem. The references indicate the entries were generated from the same lab and that sequence B was published three years earlier than sequence A. This suggests sequence A may be a revised version of sequence B.

5. Open problems.

We have not attempted a systematic evaluation of the biological significance of alignments of the scale computable with *sim2*. Such a study would be both arduous and interesting.

Schöniger and Waterman (1992) present a time-efficient algorithm for computing *n* best non-intersecting alignments with *inversions* (reverse complements). Perhaps some adaptation of their ideas would allow *sim2* to use inversions.

Acknowledgements

We benefited from a number of stimulating discussions with David Lipman. K.-M. C. and W. M. are supported by grant R01 LM05110 from the National Library of Medicine.

References

Altschul, S., W. Gish, W. Miller, E. Myers and D. Lipman (1990) A basic local alignment search tool. *J. Mol. Biol.* **215**, 403-410.

Blattner,F. R., V. D. Burland, G. Plunkett III, H. J. Sofia and D. L. Daniels (1993) Analysis of the *Escherichia coli* genome IV. DNA sequence of the region from 89.2 to 92.8 minutes. *Nucleic Acids Res.* **21**, 5408-5417.

Burland, V., G. Plunkett, D. L. Daniels and F. R. Blattner (1993) DNA sequence and analysis of 136 kilobases of the *Escherichia coli* genome: organizational symmetry around the origin of replication. *Genomics* **16**, 551-61.

Chao, K.-M., R. C. Hardison and W. Miller (1993) Constrained sequence alignment. *Bull. Math. Biol.* **55**, 503-524.

Chao, K.-M. and W. Miller (1994) Linear-space algorithms that build local alignments from fragments. *Algorithmica*, in press.

Daniels, D. L., G. Plunkett, V. Burland and F. R. Blattner (1992) Analysis of the *Escherichia coli* genome: DNA sequence of the region from 84.5 to 86.5 minutes. *Science* **257**, 771-778.

Eppstein, D., Z. Galil, R. Giancarlo and G. F. Italiano (1992) Sparse dynamic programming. I: linear cost functions. *J. Assoc. Comput. Mach.* **39**, 519-545.

Hirschberg, D. S. (1975) A linear space algorithm for computing maximal common subsequences. *Comm. Assoc. Comput. Mach.* **18**, 341-343.

Huang, X., R. C. Hardison and W. Miller (1990) A space-efficient algorithm for local similarities. *CABIOS* **6**, 373-381.

Pearson, W. R. and D. Lipman (1988) Improved tools for biological sequence comparison. *Proc. Natl. Acad. Sci USA* **85**, 2444-2448.

Plunkett, G., V. Burland, D. L. Daniels and F. R. Blattner (1993) Analysis of the *Escherichia coli* genome. III. DNA sequence of the region from 87.2 to 89.2 minutes. *Nucleic Acids Res* **21**, 3391-3398.

Pugh, W. (1990) Skip lists: a probabilistic alternative to balanced trees. *Comm. Assoc. Comput. Mach.* **33**, 668-676.

Rudd, K. E. (1993) Maps, genes, sequences, and computers: an *Escherichia coli* case study. *ASM News* **59**, 335-342.

Schöniger, M. and Waterman M. (1992) A local algorithm for DNA sequence alignment with inversions. *Bull. Math. Biol.* **54**, 521-536.

Smith, T. F. and M. S. Waterman (1981) Identification of common molecular sequences. *J. Mol. Biol.* **147**, 195-197.

Sofia, H. J., V. Burland,, D. L. Daniels, G. Plunkett III and F. R. Blattner (1994) Analysis of the *Escherichia coli* genome. V. DNA sequence of the region from 76.0 to 81.5 minutes. Manuscript.

Waterman, M. S. and M. Eggert (1987) A new algorithm for best subsequence alignments with application to tRNA-rRNA comparisons. *J. Mol. Biol.* **197**, 723-728.

Wilbur, W. and D. Lipman (1983) Rapid similarity searches of nucleic acid and protein data banks. *Proc. Nat. Acad. Sci. USA* **80**, 726-730.

Wilbur, W. and D. Lipman (1984) The context dependent comparison of biological sequences. *SIAM J. Appl. Math.* **44**, 557-567.

Zhang, J., J. Ostell and K. Rudd (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**, 58-67.