Special Issue Papers

Serafim Batzoglou
received his PhD from MIT in 2000. He is an Assistant Professor of Computer Science at Stanford, and his research focus is computational biology.

The many faces of sequence alignment

Serafim Batzoglou

Date received (in revised form): 3rd December 2004

Keywords: *sequence alignment, local alignment, multiple alignment, synteny detection, rearrangements, hidden Markov model*

Abstract

Starting with the sequencing of the mouse genome in 2002, we have entered a period where the main focus of genomics will be to compare multiple genomes in order to learn about human biology and evolution at the DNA level. Alignment methods are the main computational component of this endeavour. This short review aims to summarise the current status of research in alignments, emphasising large-scale genomic comparisons and suggesting possible directions that will be explored in the near future.

INTRODUCTION

Sequence alignment is the poster child of bioinformatics. Alignment is the most basic component of biological sequence manipulation, and has diverse applications in sequence assembly, sequence annotation, structural and functional predictions for genes and proteins, phylogeny and evolutionary analysis. The computational problem of sequence alignment was born perhaps in 1966 with the definition of the *edit distance* between two strings as the minimum number of edit operations – insertions, deletions and letter substitutions – needed to transform one string into another.[1] The subsequent literature on alignment has been enormous, and includes seminal papers such as the original Needleman–Wunsch dynamic programming solution,[2] the Smith–Waterman algorithm for local alignment,[3] the introduction of affine gaps,[4] the progressive approach to multiple alignment[5] and the BLAST tool[6] that enabled genome-scale similarity search. Recently, the literature on basic methodology and tools development has been growing rather than shrinking, indicating that the alignment problem is still not solved. How can that be, after nearly 40 years of research and literally hundreds of available tools?

There are two main reasons that alignments should remain an open problem. First, and most important, alignment is not a single problem but rather a collection of many quite diverse questions that all have in common the search for sequence similarity. Starting from the definition of alignment, there are two biologically meaningful formulations – one based on the desire to find evolutionary relationships and one based on the desire to find putative functional relationships. Given some biological sequences, the first formulation may suggest asking for a mapping (an edge) between every pair of letters that are derived from the same ancestral letter through the replication machinery of cells. This definition, however, is both too restrictive by disallowing point mutations or convergent evolution, and too permissive by not seeking to find the truly orthologous segments between the sequences of two species. The second formulation may seek to find all sequence similarities that are more significant than a threshold above random similarity, implying some common function.

Serafim Batzoglou,
Department of Computer Science,
Stanford University,
James H. Clark Center,
318 Campus Drive, RM S-266,
Stanford, CA 94305–5428, USA

E-mail: serafim@cs.stanford.edu

However, this formulation needs a model by which 'random' and 'non-random' sequences are generated – coming back to the almost impossible challenge of modelling sequence evolution comprehensively. Recognising that a single definition of alignment is not warranted, researchers have formulated a large number of diverse alignment problems that address different scientific goals. Such formulations include *local alignment*, *global alignment*, *synteny detection*, *multiple alignment*, alignment of proteins, nucleotides or non-coding RNA structures, and many other variants. A second factor that keeps the alignment problem alive is the exponential expansion of biological sequence databases – faster than Moore's law. This is markedly true today, when institutions such as the NIH have recognised the importance of comparative genomics to interpreting the human genome and are sequencing several whole mammalian genomes whose main purpose will be alignment to human.[7] The premise of these efforts is that by comparing several genomes, we will learn about human biology: regions that are biologically important tend to be more constrained by evolution and therefore more conserved than average. Moreover, the pattern of sequence conservation can hint at the specific function of a region.

There are several formulations of similarity search in biosequences, such as local, global and multiple alignment, and synteny mapping

A brief description follows of the different flavours of alignments. Given two sequences x and y, the *global alignment* problem asks for the optimal transformation of one sequence into the other with a series of *edit* operations – substitutions of one letter for another, or deletions or insertions of a substring. The objective function is usually linear in the (weighted) number of substitutions, with $s(a, b)$ given by a substitution matrix for any pair of letters a and b, and affine in the gaps, with an insertion or deletion of length l being penalised by $c_1 + c_2 l$. The *local alignment* problem asks for all sufficiently strong local similarities between two sequences to be found; the famous Smith–Waterman algorithm finds the best-scoring local alignment in time proportional to the product of the lengths of two sequences, and was later extended to find all non-intersecting local alignments that have a score greater than a threshold.[8] [The Waterman–Eggert algorithm[8] defines as intersecting two alignments that share a letter pairing (x_i, y_j) (either a match or a mismatch), and finds non-intersecting local alignments in decreasing order of their score, at an additional computational cost. A more common approach is to use directly the dynamic programming table constructed from the original Smith–Waterman algorithm, and construct local alignments by greedily using a traceback procedure starting from high-scoring cells in the table.[9]]

The BLAST algorithm and similar methods that were developed later approach this problem heuristically with subquadratic computation time in practice. *Synteny detection* is the problem of finding all sufficiently similar regions between a pair of genomes. Synteny detection differs from local alignment in the filtering criterion: whereas local aligners find all sufficiently similar pairs of substrings in the two genomes (and because of repeats and duplications they usually return too much 'junk' homology), the goal of synteny detection is typically to find all true *orthologues*, which are pairs of regions that evolved from the same region in the immediate ancestor of the two species. *Multiple alignment* is the problem of finding the 'best' way to arrange the letters of several sequences into columns that are composed of letters and gaps, in order to reveal the similarity between the sequences. This generalisation is significantly more complicated, and even defining an objective scoring function for multiple alignment is very much an open problem. Multiple alignments are usually constructed and scored by decomposing the problem into many pairwise alignments.

The type of biological sequences in question matters in the selection of objective functions and techniques. Proteins tend to come in large families of related short protein sequences, which are composed of 20 amino acids and typically are very dissimilar to one another: fewer than 25 per cent of the letters may match between two related proteins. Genomes on the other hand, are extremely long – in the case of a mammal usually about 3 Gb long – and very dissimilar on average, but contain islands of high sequence identity, such as genes. Such features dictate distinct algorithms that are practical and accurate in each case.

In the following sections, the different forms of alignment problems are discussed, with emphasis on some of the latest research in genomic sequence comparison. A list of useful web resources is given in Table 1.

Algorithms based on search for similar words, such as FASTA and BLAST, have enabled fast homology search in large sequence databases

LOCAL ALIGNMENT

Local alignment is perhaps the most straightforward way to compare two sequences – the method that makes the fewest assumptions about how the similarity should be organised: one is asked to find all subsequences that have similarity higher than a threshold. This definition works reasonably well in finding all pairs of genes or evolutionarily constrained elements between two genomes, transposons and other repeats, and any other similarities. Local similarity search is robust with respect to rearrangements (or *shuffles*) between genomes, such as inversions, translocations or duplications.

The most widely used bioinformatics tools in genomics have been index-based local aligners starting with BLAST. These tools assume that a local alignment of interest contains a match with score greater than a threshold between two constant-sized words (Figure 1A). The match is usually exact in nucleotide searches, whereas inexact matches are useful in finding protein homology because some amino acids tend to be substituted with others often. The original BLAST tool found all such matches rapidly by forming an index (a dictionary) of all the *k*-long words (*k*-mers) in a query, and searching in linear time a database for matches to those words. Every match would initiate an extension to the left and to the right, to deduce if that match is contained in a sufficiently strong local alignment (Figure 1B). The memory required for this procedure is proportional to the length of the query, and the time is roughly proportional to the number of matches that initiate extensions. In practice, if *k* is large enough (and especially if the sequences are masked for repeats), not too many matches exist and the search tends to complete rapidly.

Advances in index-based local alignment search

The index-based alignment method has revolutionised sequence comparison, and

Table 1: Web resources

Website	Description
http://www.ncbi.nlm.nih.gov/BLAST	BLAST server at NCBI
http://blast.wustl.edu	WU-BLAST – a really fast version of BLAST
http://pipmaker.bx.psu.edu/pipmaker	PipMaker alignments and percentage identity plots
http://genome.ucsc.edu	UCSC genome browser
http://www.ensembl.org	ENSEMBL genome browser
http://pipeline.lbl.gov/cgi-bin/gateway2	VISTA genome browser
http://lagan.stanford.edu	LAGAN toolkit for genomic alignment
http://baboon.math.berkeley.edu/mavid	AVID toolkit for genomic alignment
http://www.cs.ucsd.edu/groups/bioinformatics/GRIMM	GRIMM rearrangement analysis tool
http://www.ebi.ac.uk/clustalw	CLUSTALW protein aligner
http://www.drive5.com/muscle	MUSCLE protein aligner
http://probcons.stanford.edu	ProbCons protein aligner

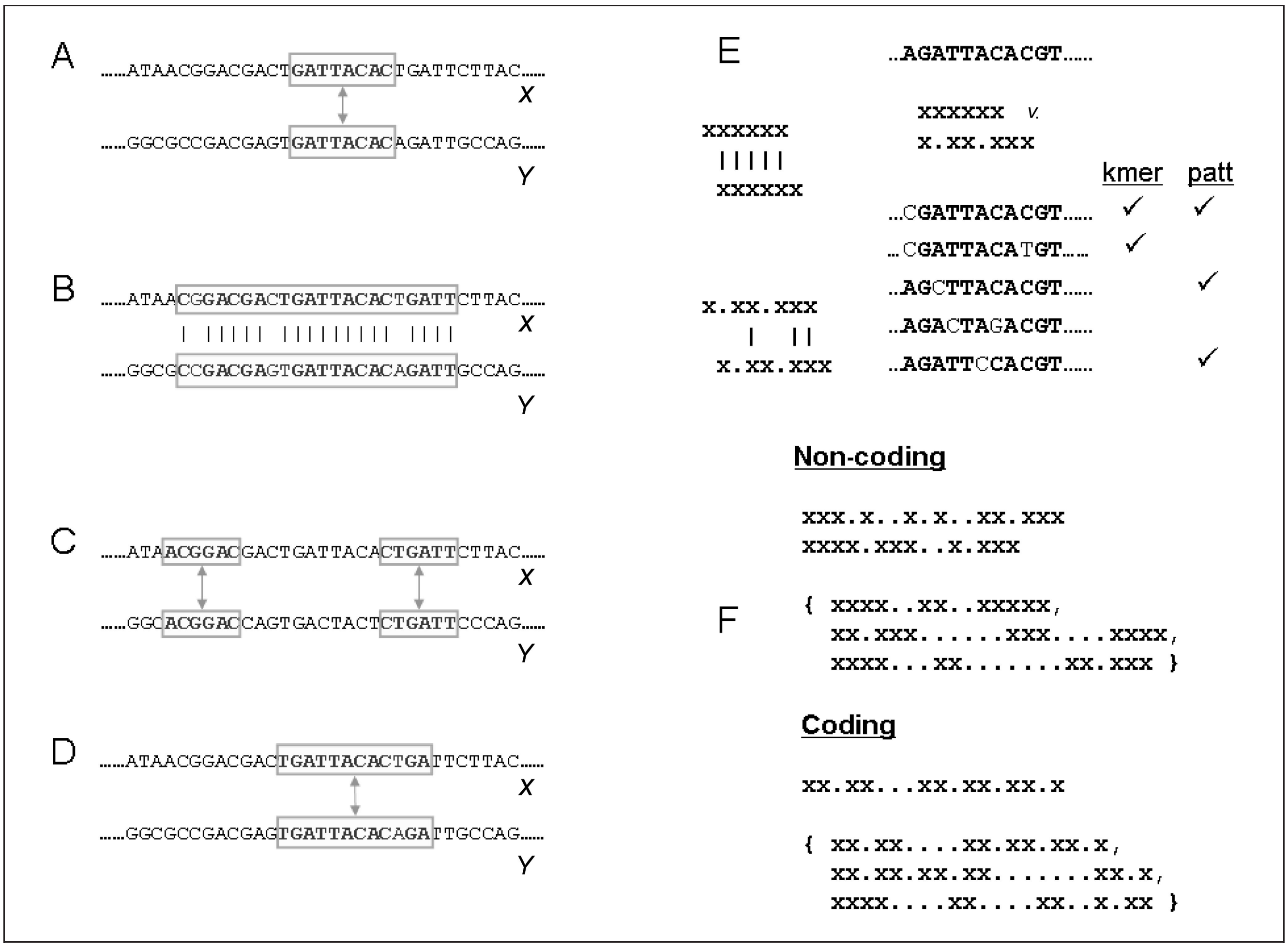

Figure 1: *Index-based local alignment.* (A) The BLAST-like technique for finding local homologies is based on first finding an exact word match (more generally, a high-scoring word match) between the two sequences. (B) All word matches are extended to the left and to the right until sequence similarity drops. The resulting local homologies that are stronger than a cutoff are reported. (C) One way to make the BLAST-like approach more sensitive (ie missing fewer sufficiently strong local homologies) without making it less specific, is to decrease the word length but to require two word matches in the same shift (or nearby shifts) with respect to one another. (D) Another way to increase the sensitivity/specificity trade-off is to require an inexact match (a match with at most one letter difference) between longer words. (E) Perhaps the most effective way is to use patterns instead of words. A word of length 6 (a 6-mer) samples five identical positions when compared with a shift of one position of itself. A pattern of length 8 with six significant positions can be constructed so as to sample only three identical positions with its shift of one position. As a result, within a highly conserved region, roughly five out of six times a failed hit in a given position (eg AGATTA *v.* the sequences shown) results in a failed hit in the next position for a 6-mer; the same failure results in only roughly three out of six times in a failure in the next position when using a pattern. (As sequence conservation drops, this difference becomes less dramatic, and gaps further complicate the performance trade-offs.) (F) Some good individual patterns or sets of three simultaneous patterns, for similarity search in non-coding or in protein coding positions, as suggested by rigorous computational analyses[18,19]

has led to an extensive literature and numerous tools with the aim of performing such searches more effectively in the general case, or in specialised settings. One significant feature of this method is the trade-off between speed and sensitivity in detecting homology. This trade-off is controlled by the word length *k*, and to a lesser degree by the hit extension method, where gapless extensions were used initially, but where gapped extensions introduced later[10] led to improvements in sensitivity. The word length *k*, however, is most significant: a shorter word length makes it more likely to detect a significant local alignment that exists between the query and database, but increases the number of matches expected to occur at random and results in costly examination (alignment extension) of all the spurious matches in order to select the significant homologies. For example, given random sequences, and given a gapless alignment of length 500 at 81 per cent sequence identity, a matching k-mer of length $k = 8$ is expected to occur within the alignment 91.5 per cent of the times, and therefore the majority of such alignments will be detected by an indexing scheme that uses 8-mers. However, the same 500-long region will have an 8-mer match with 3 million locations on average just by chance in a random genome of length 3 Gb. In contrast, 14-mer matches will generate just 399 matches by chance, but will detect only 31.4 per cent of 500-long local alignments at 81 per cent sequence identity.[11] Three of the techniques for increasing the ratio of sensitivity (detection of strong local alignments) to specificity (avoidance of spurious word matches in regions that do not align), are:

Several tricks can improve the sensitivity and specificity of indexing a sequence database

- requiring two words at a maximum distance to one another, instead of a single word, to initiate a match (Figure 1C);
- matching of a k-mer of non-consecutive positions (a *pattern*), which is a strategy used in PatternHunter;[12] and
- matching of inexact k-mers (Figure 1D).

The first technique is employed by most local aligners, such as several BLAST variants (NCBI BLAST,[10] WU-BLAST (a very efficient variant or BLAST[13]); BLASTZ,[14] BLAT[11] and PatternHunter). CHAOS[15] uses a variant in which chains of seeds are first constructed, and only sufficiently strong chains are further examined. PatternHunter and BLASTZ use the second technique, while CHAOS uses the third technique.

The pattern-based approach employed in PatternHunter is highly effective in finding at least one match between two homologous regions. To see why this is true, imagine that regions $x_1 \ldots x_n$ and $y_1 \ldots y_n$ are homologous with no gaps. A k-mer samples positions $x_i \ldots x_{i+k-1}$ that should match to $y_i \ldots y_{i+k-1}$. If they do not match, the next k-mer shift, $x_{i+1} \ldots x_{i+k}$, samples $k - 1$ identical positions and only one different position; the only way it can match is if the sole mismatch was at x_i. This observation motivates the indexing of patterns, which are k non-consecutive positions: a pattern indexed at positions i has an overlap of fewer than $k - 1$ letters with the same pattern indexed at position $i + 1$, and therefore a failure at i is less correlated with a failure at $i + 1$ (Figure 1E). Effectively, the pattern-based approach keeps the expected number of matches between x and y constant (actually, this quantity decreases slightly because patterns are longer than k-mers and so fewer samples fit within $|x|$ positions), but increases the chance that *at least one match* is detected. Given just one match at the right place, BLAST-like techniques extend it and detect homology. This clever approach has a rather long history, starting with FLASH[16] but has received more attention by the alignment community recently. PatternHunter has been used in whole-genome comparative analyses.[17] Methods that find the most effective patterns were devised[18] (Figure 1F). An additional extension of this idea is to index multiple different patterns per position,[19] increasing sensitivity. In general, pattern-based indexing is significantly better than k-mer indexing in most cases. Benefits drop when the homology to be detected is very short (because then the number of words sampled is significantly smaller with long patterns), or contains frequent gaps (because patterns are longer and therefore a match is more likely to be interrupted by a gap).

An interesting problem, given the rapid rise of available data in the form of multiple alignments, is to extend pairwise local alignment to the comparison of a sequence to a multiple alignment, or profile. There are several sequence-to-profile alignment methods, of which the best known is PSI-BLAST.[10] Such methods are employed with great success in distant protein homology search, for example for protein structural prediction.[20,21] None of them, however, focuses on high-throughput comparison of queries to a mammalian-size reference alignment.

GLOBAL ALIGNMENT

The standard edit distance formulation of sequence alignment leads to quadratic-time dynamic programming algorithms

A global alignment between two strings $x = x_1 \ldots x_m$ and $y = y_1 \ldots y_n$ is a placement of gaps (or '–' characters) at the beginning or end, or between any pair of letters of the strings, so that the resulting padded strings are of equal length, and pairs of gaps that map to one another are disallowed. An equivalent formulation involves a matrix of dimension $(m + 1) \times (n + 1)$ where in the x-axis lies the sequence $0x_1 \ldots x_m$ from left to right, in the y-axis lies the sequence $0y_1 \ldots y_n$ from top to bottom, and an alignment is any path from (0,0) to (m,n) using only *down* (+0, +1), *right* (+1, +0) and *down–right* (+1, +1) moves, corresponding to gaps in x, gaps in y and mappings between pairs of letters, respectively. Sometimes an additional restriction applies, of never following *down* with *right*, or *right* with *down*, leading to slightly faster algorithms. Yet another equivalent formulation uses a finite automaton with three states (Figure 2A) that generate two sequences simultaneously: state M emits the next two symbols x_i and y_j, state I emits the next symbol x_i, and state J emits the next symbol y_j. A sequence of states such that the number of M plus I states equals m, and the number of M plus J states equals n, corresponds to a global alignment of x and y. (Transitions between I and J would correspond to gaps in one sequence followed by gaps in the other, and are often omitted.)

One of the earliest global alignment methods was the Needleman–Wunsch dynamic programming algorithm that computes the optimal edit distance between two strings.[2] The original procedure required cubic time and is not used in practice but simple dynamic programming procedures (see for instance Gusfield[9]) require time and space proportional to the product of the lengths of the two sequences. Later, algorithms were devised that use only linear space.[23,24] The Smith–Waterman algorithm[3] computes the best *local* alignment between two strings, which is defined as the best global alignment of any two substrings of the strings. Another important extension was the introduction of a more elaborate gap model: gap lengths in alignments of actual biological sequences do not follow a geometric distribution but rather have a tendency to be several nucleotides long. The *affine* gap model scores gaps so as to reflect this biological feature, penalising a gap of length l with a constant C_{open} plus a linear term $l \times C_{\text{extend}}$.[4] Transitions in the 3-state finite automaton can be labelled with scores reflecting edit-distance scoring parameters (Figure 2A), so that the best-scoring state path for two sequences corresponds to their optimal alignment under affine gaps.

Most of the early alignment algorithms were designed primarily for application to protein sequences. The recent proliferation of available DNA sequences and complete genomes of related organisms underlined two problems with methods based on dynamic-programming computation of edit distance. First, such methods run in quadratic time in the lengths of the sequences, and therefore are too inefficient for application to long genomic sequences. Second, genomic sequences of related organisms are often similar in short, functionally important regions that are located in the midst of much longer regions of little similarity. The affine gap model is especially unsuccessful in correctly aligning short conserved regions because the positive score gained by similarity within such regions cannot counterbalance the

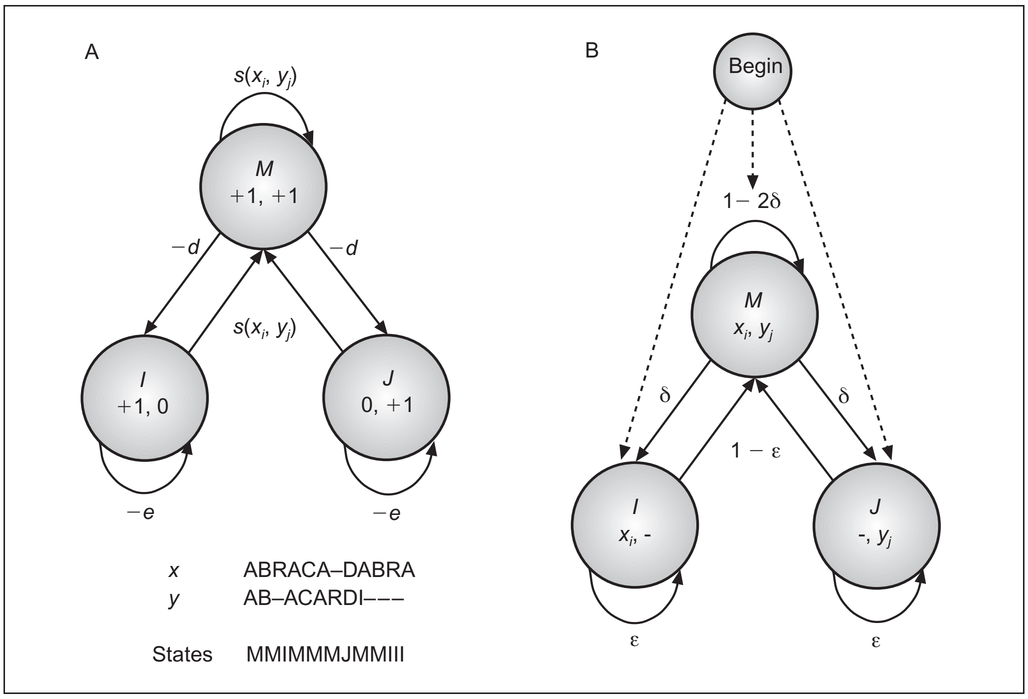

Figure 2: *A finite automaton for global alignment.* (A) The state diagram of a 3-state automaton that scores an alignment under the (single) affine gap model. State *M* emits two letters, x_i and y_j, and advances to positions $i + 1$ and $j + 1$. State *I* emits symbol x_i and a gap, and advances to the next position $i + 1$ in *x*. Similarly for state *J*. Gaps are opened at every transition from *M* to *I* or *J*, and incur a gap penalty equal to $d + el$, where *l* is the length of the gap. Letter matches and mismatches are scored according to the substitution function *s*. (B) The finite automaton can be transformed into an equivalent hidden Markov model (HMM) with a transformation of scoring parameters into log-odds ratios.[22] Intuitively, δ controls the gap initiation penalty, ε controls the gap extension penalty (the precise relation between these values and *d*, *e* is somewhat complicated), and *M* emits symbols with substitution scores that reflect the log-likelihood ratio of two letters being aligned in related sequences, versus each letter occurring independently. The Viterbi algorithm for this HMM produces the most likely alignment, which is equivalent to the highest-scoring alignment of the automaton in (A)

negative score caused by the long flanking gaps. This challenge reflects the fact that affine gaps, like linear gaps, still imply a geometric distribution of gap lengths, while in biological reality longer gaps occur much more frequently, notably because of insertions of transposable elements. Double-affine gaps, corresponding to a 5-state finite automaton that includes 'long' gap states I' and J', or logarithmic gaps and other more involved models are meaningful but less commonly used.

Fast global aligners are based on chaining of local alignments

Fast global alignment

The shortcomings in computational efficiency of classical methods motivated the recent development of several efficient (sub-quadratic) global alignment algorithms designed for application to distantly related sequences. The PipMaker system[25] based on a gapped version of BLAST[10] finds and visualises a set of significant local alignments between a pair of long genomic sequences. Several global alignment systems run in subquadratic time: MUMmer,[26] DBA,[27] GLASS,[28] WABA,[29] AVID[30] and LAGAN,[31] among many others. Most of these methods are based on a *chaining* strategy of first finding and chaining words or local alignments between the two sequences,[9] thereby creating a set of anchors; these anchors cut the large alignment problem into many smaller ones. The basic idea is illustrated in Figure 3. Chaining works

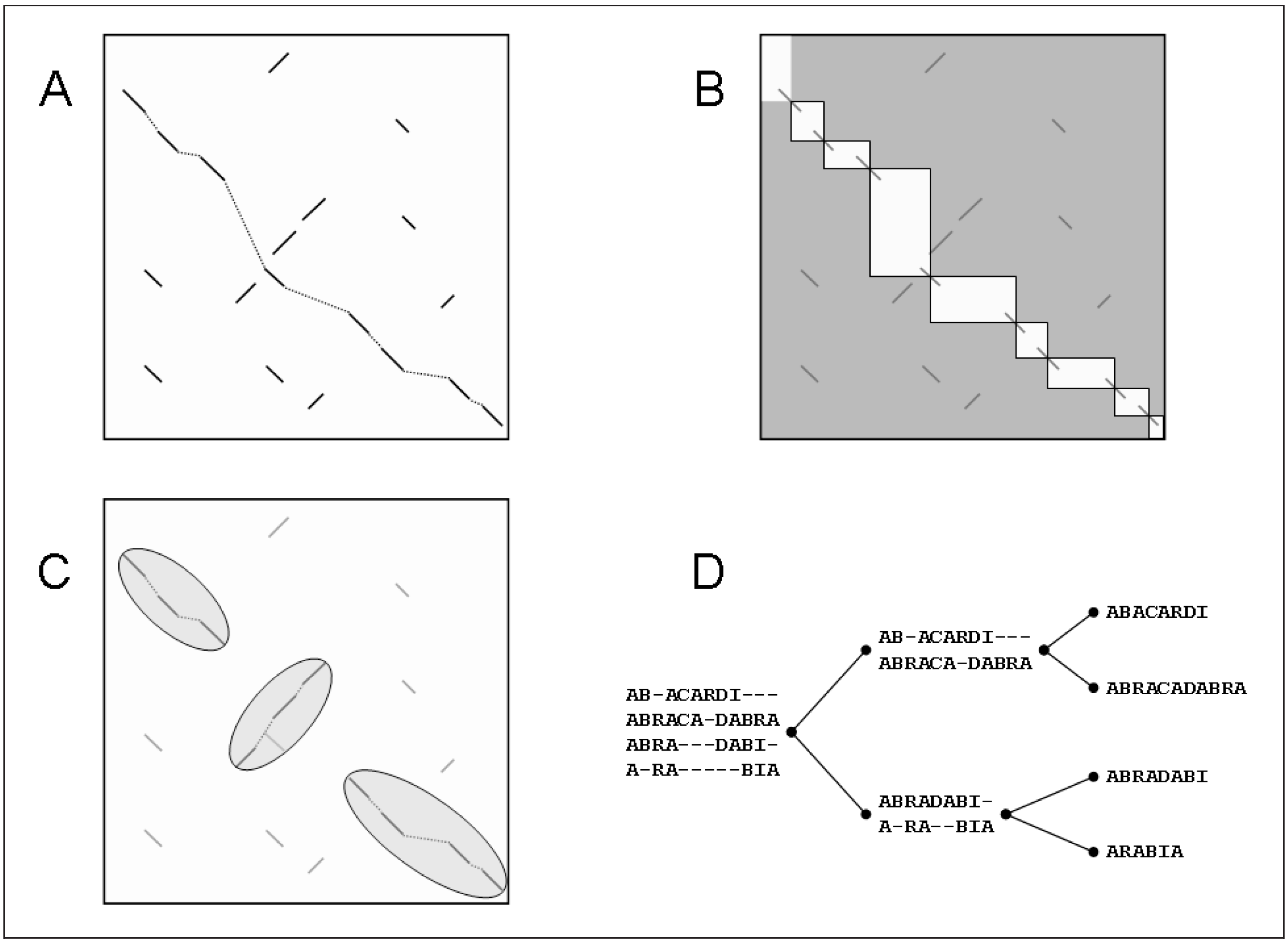

Figure 3: *High-throughput global alignment.* (A) Methods that align long genomic sequences usually rely on first quickly finding a set of local alignments between two sequences, and then constructing a monotonic chain of those alignments. (B) After the chain is constructed, most methods align the regions in between; the original global alignment problem has been effectively cut into many smaller ones and most of the search space is excluded. (C) Global aligners that follow the methodology outlined in (A) and (B) miss rearrangements such as inversions that are common within megabase-long orthologous regions in mammals. More recent methods, such as SLAGAN or CHAINNET, heuristically pick such events while still filtering out the majority of spurious local alignments. (D) To construct an alignment between multiple sequences, practical systems use the progressive strategy of constructing partial alignments, or profiles, in the order of a tree connecting the sequences. The tree can follow the phylogenetic relationship of the sequences, or simply be a hierarchical clustering according to similarity – the second method may produce more accurate alignments in practice, because it performs the easier alignments first, thus propagating fewer errors to later steps

well in practice in addressing both shortcomings of pure dynamic-programming approaches, namely insufficient speed and intolerance of long gaps. Concerning speed, chaining relies on three steps: (1) exact word matching or index-based local alignment, which is much faster than the time required for full dynamic programming; (2) construction of the optimal chain of local hits, a procedure that can be done in time 0(n log n) where n is the number of hits, using an extension of the Longest Increasing Subsequence algorithm; and (3) potential computation of alignments in the regions in between the chained pieces, which give rise to much smaller alignment problems. Concerning accuracy in dealing with long gaps, chaining relies precisely on the fact that DNA sequence similarity comes in islands of highly conserved (evolutionarily constrained) regions, flanked by less conserved pieces that are typically much

Local alignments can be chained in time O(n log n) using an extension of the longest increasing subsequence algorithm

longer in higher organisms. This property makes it reasonable to first find the highly similar regions and construct a map based on them, and only later deal with the potentially unalignable regions in-between. Chaining-based global aligners do not suffer from the shortcomings of the affine gap function. Instead, by their hierarchical nature they impose a two- or multi-level gap penalty where long gaps are encouraged in order to detect short homologies.

Synteny mapping, in its most general form, is a filtering criterion over all local similarities

SYNTENY MAPPING AND REARRANGEMENTS

While local alignment is a way to compare two genomes that does not make any assumptions about the specific evolutionary relationship between them, synteny detection is an organisation of local alignments into a coherent global picture. The aim is to map the orthologous blocks – the regions that are assumed to be derived from the same region in the genome of the immediate ancestor of the two species. Early on the notion of orthology mapping was introduced,[32] and it was estimated that there are roughly 180 *conserved segments* – pairs of regions with conserved gene order without rearrangements – between the human and mouse genomes. This notion was later generalised to *synteny blocks*, which are regions that can be converted into conserved segments by local rearrangements, the *micro-rearrangements*.[33] Moreover, with the abundance of genomic data it became clear that in addition to genes, conserved non-coding regions are important for detecting and comparing synteny blocks.

Approaches that attempt to map rearrangements are more ambitious, with the ultimate goal of reconstructing the order of immediate ancestor of two species

Synteny detection would be easy were it not for the abundance of local alignments between non-orthologous locations. Such alignments are mainly the result of segmental duplications and repeats that copy regions within a genome. These events, coupled with gene loss and sequence divergence in general, deletions, and genomic rearrangements of various sizes from the tiny reversals of a few hundred nucleotides to whole-chromosome fissions and fusions, make synteny detection challenging but necessary, because otherwise the picture of orthology between two moderately distant genomes is cluttered with an enormous amount of local alignment noise.

Many approaches exist for synteny mapping, and most are based on the same general scheme of first obtaining a set of local alignments between two genomes, and then grouping those alignments into clusters so that each spans a specific region of a chromosome in each species. The comparison of human and mouse was a first opportunity for large-scale synteny mapping, and three methodologies were introduced in this context.[17,33,34] In the first approach, a local aligner was first applied (PatternHunter) to find more than 0.5 M bidirectional-best local similarities (coding and non-coding), the *anchors*; those were grouped into maximal *syntenic segments* whenever they occurred in the same pair of chromosomes, in the same order and orientation; finally, those segments were grouped into larger *syntenic blocks* by concatenating adjacent segments that may be shuffled with respect to one another.[17] In the second approach, the *GRIMM-synteny algorithm*, the same anchors were arranged in a graph where two anchors were connected by an edge if their distance was smaller than a threshold (the Manhattan distance was used, which is the sum of distances in each genome); then, synteny blocks were formed according to the connected components of this graph that spanned regions longer than a threshold.[33] The third approach used a different local aligner for speed (BLAT) to form anchors, grouped anchors according to proximity and same order and orientation, and mapped each mouse contig to human according to the strongest group of anchors. All methods largely agreed on the long syntenic blocks, and differed on their ability to handle local shuffles, and on the resolution of synteny detected. As was pointed out,[33] as the length of a syntenic block decreases, the block becomes less reliable because it may be the result of spurious local alignments and sequencing errors.

The rat genome[35] provided an opportunity to perform mammalian-scale synteny detection across multiple genomes – in this case human, mouse and rat. The GRIMM methodology was extended to this setting, to find all pairwise and three-way blocks, and to reconstruct a putative ancestral rodent.[36] Additional methods that were developed in that context include Pash, a very efficient and parallelisable method that finds synteny between two regions by forming groups of *k*-mers that lie on the same diagonal of the alignment matrix of the two sequences, implying the same 'shift' between the two genomes,[37] and a progressive local/global technique based on first finding syntenic mouse-rat blocks, and then mapping those to human.[38]

Recently it was realized that microarrangements are abundant in genomic DNA

Micro-rearrangements complicate the alignment task within a syntenic block. They have been observed to be abundant between species such as human and mouse[33] and therefore cannot be ignored during global alignment. The concept of *glocal* alignment has been introduced[39] to provide a happy median between an unfiltered collection of all local alignments between two regions, and a strict global alignment that would miss events such as local inversions (Figure 3C) and transpositions. A glocal map is a chain of local alignments that is *monotonic* in one sequence so that the coordinates of successive local alignments are increasing, while it can jump around in the second sequence to accommodate rearrangements. The first sequence is typically the more interesting, or better-quality sequence. Shuffle-LAGAN[39] computes such chains in O(n log n) time where n is the number of local alignments, by an extension of the Eppstein–Galil algorithm,[40] and subsequently performs global alignment on the consistent parts of the chain. This procedure unfortunately does not properly model the sizes or end-points of rearrangement events – conceivably, a stochastic pair-CFG (context-free grammar):[22] could be used for that purpose, under the strong assumption that rearrangement events are nested, but that would result in very slow parsing algorithms. The Shuffle methodology was later extended to perform alignment of draft-against-finished sequence, and as a basis for global synteny detection between multiple genomes.[41] Another approach that provides filtered sets of local alignments in a global map, while respecting local rearrangements, is the chain-and-net approach.[42] This approach is hierarchical, which makes sense biologically because rearrangements most likely happen in a hierarchical fashion from very short inversions of a few hundred nucleotides, up to whole-chromosome fusions and fissions. The program CHAINNET forms chains of local alignments that are ordered in both sequences; then, it picks chains iteratively and adds them to the global map, each time making sure to throw out the parts of a chain that intersect with nucleotides already covered by previously accepted chains in a genome of interest (eg human). This way, each nucleotide of the genome of interest is covered by at most one nucleotide of the second genome, but shuffles of any size are accommodated. Based on this method, an analysis of micro-rearrangements within the mouse genome relative to human revealed a very high frequency of such events. For example, two inversions per Mb of mouse genome were detected (excluding combinations of inversion and duplication), of median size 814 bp.[42] It is important to note that the regional frequency of such events varies considerably across the genomes.

MULTIPLE SEQUENCE ALIGNMENT

Multiple sequence alignments are a natural extension of two-sequence comparison, and a powerful way to study biological sequences. They are essential for the computation of local rates of evolution, giving a quantitative measure of the strength of evolutionary constraints and the functional importance of local regions.[43,44] Even weak similarity across several sequences reveals important conserved biological features.[45] Recently, several methods have emerged for detecting regions that evolve slower than neutral on the basis of a multiple genomic sequence alignment.[46–48]

Progressive alignment

Multiple alignment tools usually decompose the problem into several pairwise alignment steps

Multiple sequence alignments are considerably more difficult to compute than pairwise alignments. For straightforward dynamic programming solutions, each additional sequence multiplies the time and memory required to compute the optimal alignment by a factor proportional to the length of the sequence. Formally, the problem is NP-complete.[49] For that reason heuristic approaches are usually applied, of which the most widely used is *progressive alignment*. This approach works by successively constructing pairwise alignments,[5,50–52] providing at the same time a hierarchical clustering of the sequences (Figure 3D). At each progressive alignment step, two multiple alignments, which are typically compressed as profiles, are aligned by a procedure that treats each as a sequence and therefore inserts entire gap columns between two columns (Figure 3D). The result is a combined multiple alignment that can be passed to the next step. The best-known system based on progressive multiple alignment is perhaps CLUSTALW.[53] Other multiple alignment systems that are mostly targeting proteins or short DNA sequences, and are based on progressive alignment, include MULTALIGN,[54] MULTAL,[55] PRRP,[56] T-COFFEE,[57] MAFFT,[58] MUSCLE,[59] Align-m[60] and PROBCONS.[61]

Scoring a multiple alignment as well as detecting conserved regions within it, are open research problems

Scaleable multiple sequence alignment

Until recently, multiple global alignment of large genomic regions was challenging owing to the enormous search space involved. Methods such as CLUSTALW have been suitable for aligning proteins, but are too time-inefficient for sequences longer than a few thousand nucleotides. In the past couple of years, several scaleable methods have been developed: MLAGAN is a method for large-scale multiple alignment based on a progressive LAGAN alignment step applied in the order of the given phylogenetic tree of the sequences.[31] MAVID is a method that can automatically construct a phylogenetic tree, and can also use gene predictions to help in constructing the chain of anchors. It uses ancestral reconstruction and therefore always aligns single ancestor sequences rather than profiles, making the method fast; however, for the same reason it does not incorporate affine gap penalties.[62] TBA is a promising alternative method that introduces the concept of a *blockset*, which is a region of similarity between a subset of the sequences, and constructs a multiple alignment composed of blocksets that are aligned with the MULTIZ local aligner.[63] The block set idea ameliorates a common problem with multiple aligners, of padding deep alignments with huge numbers of gaps in place of lineage-specific transpositions or deletions; those gaps potentially affect scoring during alignment, or subsequent interpretation of the alignment.

Scoring function

Besides efficiency, generalising the objective function from pairwise to multiple alignments is also a challenge. Two of the main scoring methods are *sum-of-pairs* and *consensus*. According to the sum-of-pairs method, the score of a multiple alignment is the (weighted) sum of scores of all the induced pairwise alignments.[64,65] In *consensus*, each position of the alignment is assigned a letter (usually the most frequent letter in that position) and the overall multiple alignment score is the sum of pairwise alignment scores of each sequence to the consensus. A weighted combination of the two methods is employed in the

ReAligner system.[66] DIALIGN[67] uses an interesting segment-to-segment objective function. A comprehensive analysis of alignment scoring functions has revealed significant differences with respect to the resulting alignment accuracy.[68]

Consistency-based scoring

DIALIGN, T-COFFEE and PROBCONS use the concept of *consistency* to improve the objective function of multiple alignment. The basic idea of consistency is that during a progressive alignment step that aligns two sequences x and y, errors can be avoided by seeking to align pairs of letters that will consistently map to the same positions of a third sequence z in subsequent alignment steps (Figure 4). Consistency was first introduced to identify anchor points for reducing the search space of a multiple alignment.[69] A mathematically elegant reformulation of consistency in terms of Boolean matrix multiplication was later given[70] and implemented in the program MALI, which builds multiple alignments from dot matrices.[71] T-COFFEE builds an alignment library by merging consistent CLUSTALW global and LALIGN[72] local pairwise alignments to form three-way alignments, which are assigned weights by percentage identity. Then, the score for aligning x_i to y_j is defined to be the sum of the weights of all alignments in the library containing that aligned residue pair.

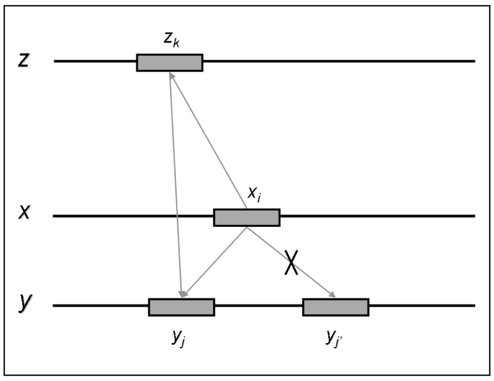

Figure 4: *Consistency in progressive alignment.* When aligning x to y, we may be in doubt as to whether to map a given letter or domain x_i to y_j or $y_{j'}$. In the context of progressive alignment that will later involve a third sequence z, we may 'ask' z whether it prefers one of the two possibilities. If x_i maps to z_k and z_k maps to y_j, the $x_i - y_j$ score should be boosted

Maximum expected accuracy

As explained,[22] the finite automaton of Figure 2, with some small technical modifications such as adding a start and a stop state, can be made into a hidden Markov model (HMM) that generates alignments. This model is called a pair-HMM because it generates a pair of sequences instead of a single one, and parameterises a probability distribution over all possible alignments of all possible pairs of sequences. The parameters of the model include emission probabilities of pairs of letters from M, or single letters from I and J, and transition probabilities between the three states. All standard HMM algorithms can be used – for example, the Viterbi procedure can compute the most likely alignment of two given sequences, and Expectation Maximisation can be used for unsupervised training of parameters. Viterbi, for instance, corresponds exactly to edit-distance dynamic programming, modulo a parameter transformation. In addition, alternative ways to find an alignment based on the HMM exist. In particular, the overall likelihood of the alignment may not be the best criterion, because this likelihood is bound to be extremely low – there is little chance to uncover the 'true' alignment between two sequences. Therefore, partial credit may be warranted for alignments that agree with the 'true' one in most predictions of letter-to-letter mappings. It has been proposed[22,73] to use the criterion of *maximum expected accuracy* to generate an alignment, where accuracy is defined as the number of correct letter-to-letter correspondences. This can be done via dynamic programming with similar time complexity as the standard Viterbi or dynamic programming procedures (in practice, about three times slower). PROBCONS adopted this objective function, and demonstrated that such alignments are significantly more accurate in protein comparison than standard methods. In addition, PROBCONS applied a *probabilistic consistency* transformation that intuitively re-estimates the probability that two letters x_i and y_j match, by heuristically summing over all possible third letters z_k in a third sequence, the posterior probabilities that triplets x_i, y_j and z_k match, given that x, y and z align. This transformation is inspired by the consistency methods that appear in programs such as T-COFFEE, and increases accuracy further. These ideas have not been applied yet to large-scale nucleotide alignment, or to chaining, rearrangement detection and other multisequence mapping problems.

Aligning alignments

When performing progressive alignment, one issue that is usually slipped under the rug is that the procedures used for aligning two intermediate profiles do not actually find the optimal alignment of these two profiles. Indeed, surprisingly, the problem of aligning two alignments with affine-gap sum-of-pairs scoring is NP-complete.[74,75] An exact algorithm has been proposed[75] that in practice performs much better than exponential time, but is still unfortunately too slow for genome-wide use. Most of the practical multiple aligners use local heuristics to estimate the gap-opening counts. Such heuristics often result in obviously wrong behaviour (ie alignments where a human eye can spot funny-looking gap placements); an exposition of different approaches is beyond the scope of this review.

Block-based formulations

Recognising that the definition of a global multiple alignment as a monotonic map between all sequences is limited, richer formulations of alignment have emerged recently. The *partial order alignment* (POA) approach models a multiple global alignment as a directed acyclic graph that includes blocks of similarity and regions of non-homology.[75] The *threaded block set* approach, which is partially implemented in the TBA program, defines a multiple alignment to be a set of blocks of similarity between subsets of the sequences that are ordered differently (threaded) according to each sequence.[63] The *A-Brujin graph* (ABA) approach further generalises POA to allow an alignment to be represented by a graph that can include cycles, allowing for the inclusion of repeated structures such as domains within each sequence.[77] Such richer structures may complicate both the computation and the intuitive interpretation of alignments, but are better able to handle the rich organisation of homology present in biological sequences.

CONCLUSION AND FUTURE DIRECTIONS

The next few years will provide a unique challenge and opportunity to alignment research. One of the immediate goals of the genomics community is to sequence and align a large number of species in order to study their biology and evolution through comparative sequence analysis. In addition, new technologies promise extremely cheap sequencing methods that will enable us to sequence almost *everything* during the next decade. The enormity of such data will be dizzying, and alignment will be the Google™ of genomics – finding all interesting connections between diverse sequences. Resources will be developed that will enable seamless navigation through alignments of a given gene, protein, domain or other functional element, in a given phylogenetic scope such as mammals, vertebrate or eukaryotes, revealing the evolutionary history of the element – when did it arise, when did it assume biological function, in which subtrees did the function get altered or lost, what is the duplication/loss history of the element? The goal will be to trace the evolution of every single letter in our genomes, between different species as well as across human populations through history. Computational reconstruction of the genomes of ancestral species will be achieved with high accuracy on the nucleotide level,[78] bringing us one step closer to Jurassic Park as well as elucidating our evolution at the molecular level. A few years ago, several unsolved problems in alignment were proposed,[79] which fell into five topics: (1) improved pairwise alignment with a statistical basis; (2) improved alignment-based gene prediction; (3) effective multiple genomic sequence alignment; (4) better alignment browsers; and (5) rigorous methods for evaluating the accuracy of alignment. Since then lots of progress has been made, especially in directions (2), (3) and (4). Directions (1) and (5) are more resistant: sequence evolution is complicated and difficult to model probabilistically and therefore rigorous yet practical statistical methods for alignment, and objective criteria for evaluation of alignments, are really hard to obtain.

Looking to the next few years, the following are just some of the problems that are significant to address:

- As suggested,[79] methods to evaluate alignment accuracy. This goes at the core of the problem: which regions are alignable, and what is a correct alignment?

- Methods that can align neutrally evolving DNA between multiple distant species, whenever the sequences have not diverged enough in principle to look non-orthologous. Perhaps one way to achieve that will be to populate the phylogenetic tree with more and more sequences, so that distant alignments are facilitated with many intermediate, easier ones.

- A definition of *alignability* – at what point is it no longer possible to do meaningful sequence alignment. Or rather, at what point can one conclude that two sequences are no longer related?

- Aligning under different models in different parts of the sequences, according to existing annotations of features such as genes, or by using automatic methods that recognise the evolutionarily constrained, neutral and plain unalignable parts.

- A rigorous way to identify breakpoints of rearrangement events in the context of a multiple species comparison, to perform synteny detection, and to reconstruct the order of the ancestral genome. Again, a highly populated tree of sequences will make this problem easier in principle

- Methods to analyse, navigate and query multiple alignments in order to identify the alignable parts, the conserved parts, the regions and subtrees that evolve faster or slower, or other interesting features, and at different granularities.

- Databases that can store, query intelligently and augment on the fly extremely large and interconnected alignments – for instance 1,000 or more vertebrate genomes. One of the major scientific goals of the next decade is to obtain an accurate and complete library of genomic evolution in the form of multi-species alignments, which will be used by biologists and bioinformaticians to understand the history and mechanisms of life at the nucleotide level.

Acknowledgments
The author wishes to thank Chuong B. Do for helpful suggestions and edits to the document, and NSF, NIH and the Sloan Foundation for support.

References

⋆ Papers of particular interest published within the period of this review.

⋆⋆ Papers of extreme interest published within the period of this review.

1. Levenshtein, V. I. (1966), 'Binary codes capable of correcting deletions, insertions, and

reversals', *Cybernetics Control Theory*, Vol. 10, pp. 707–710.

2. Needleman, S. B. and Wunsch, C. D. (1970), 'A general method applicable to the search for similarities in the amino acid sequence of two proteins', *J. Mol. Biol.*, Vol. 48, pp. 443–453.

3. Smith, T. F. and Waterman, M. S. (1981), 'Identification of common molecular subsequences', *J. Mol. Biol.*, Vol. 147, pp. 195–197.

4. Gotoh, O. (1982), 'An improved algorithm for matching biological sequences', *J. Mol. Biol.*, Vol 162, pp. 705–708.

5. Feng, D. F. and Doolittle, R. F. (1987), 'Progressive alignment of amino acid sequences as a prerequisite to correct phylogenetic trees', *J. Mol. Evolution*, Vol. 25, pp. 351–360.

6. Altschul, S. F., Gish, W., Miller, W. *et al.* (1990), 'Basic local alignment search tool', *J. Mol. Biol.*, Vol. 215, pp. 403–410.

7. URL: www.genome.gov/12511858

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

9. Gusfield, D. (1997), 'Algorithms on Strings, Trees, and Sequences', Cambridge University Press, New York.

10. Altschul, S. F., Madden, T. L., Schoffer, A. A. *et al.* (1997), 'Gapped BLAST and PSI–BLAST: A new generation of protein database search programs', *Nucleic Acids Res.*, Vol. 25, pp. 3389–3402.

11. **Kent, W. J. (2002), 'BLAT – the BLAST-like alignment tool', *Genome Res.*, Vol. 12, pp. 656–664.

12. **Ma, B., Tromp, J. and Li, M. (2002), 'PatternHunter: Faster and more sensitive homology search', *Bioinformatics*, Vol. 18, pp. 440–445.

13. **URL: http://blast.wustl.edu/

14. **Schwartz, S., Kent, W. J., Smit, A. *et al.* (2003), 'Human–mouse alignments with BLASTZ', *Genome Res.*, Vol. 13, pp. 103–107.

15. *Brudno, M. and Morgenstern, B. (2002), 'Fast and sensitive alignment of large genomic sequences', in 'Proceedings of the Bioinformatics Conference (CSB)', IEEE Computer Society, pp. 138–147.

16. Califano, A. and Rigoutsos, I. (1993), 'FLASH: A fast look-up algorithm for string homology', in 'Proceedings of the 1st International Conference on Intelligent Systems for Molecular Biology', AAAI Press, Palo Alto, CA, pp. 56–64.

17. **Waterston, R. H., Lindblad-Toh, K., Birney, E. *et al.* (2002), 'Initial sequencing and comparative analysis of the mouse genome', *Nature*, Vol. 420, pp. 520–562.

18. **Buhler, J., Keich, U. and Sun, Y. (2003), 'Designing seeds for similarity search in genomic DNA', in 'Proceedings of the Seventh Annual International Conference on Research in Computational Molecular Biology (RECOMB)', ACM Press, New York, pp. 67–75.

19. **Sun, Y. and Buhler, J. (2004), 'Designing multiple simultaneous seeds for DNA similarity search', in 'Proceedings of the Eighth Annual International Conference on Research in Computational Molecular Biology (RECOMB)', ACM Press, New York, pp. 76–84.

20. Jones, D. T. (1999), 'Protein secondary structure prediction based on position-specific scoring matrices', *J. Mol. Biol.*, Vol. 292, pp. 195–202.

21. McGuffin, L. J. and Jones, D. T. (2003), 'Improvement of the GenTHREADER method for genomic fold recognition', *Bioinformatics*, Vol. 19, pp. 874–881.

22. Durbin, R., Eddy, S., Krogh, A. and Mitchison, G. (1997), 'Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids', Cambridge University Press, Cambridge.

23. Hirschberg, D. (1975), 'A linear space algorithm for computing maximal common subsequences', *Comm. ACM*, Vol. 18, pp. 341–343.

24. Myers, E. W. and Miller, W. (1988), 'Optimal alignments in linear space', *Computing Appl. Biosci.*, Vol. 4, pp. 11–17.

25. Schwartz, S., Zhang, Z., Frazer, K. A. *et al.* (2000), 'PipMaker – a web server for aligning two genomic DNA sequences', *Genome Res.*, Vol. 10, pp. 577–586.

26. Delcher, A. L., Kasif, S., Fleischman, R. *et al.* (1999), 'Alignment of whole genomes', *Nucleic Acids Res.*, Vol. 27, pp. 2369–2376.

27. Jareborg, N., Birney, E. and Durbin, R. (1999), 'Comparative analysis of noncoding regions of 77 orthologous mouse and human gene pairs', *Genome Res.*, Vol. 9, pp. 815–824.

28. Batzoglou, S., Pachter, L., Mesirov, J. *et al.* (2000), 'Human and mouse gene structure: Comparative analysis and application to exon prediction', *Genome Res.*, Vol. 10, pp. 950–958.

29. Kent, W. J. and Zahler, A. M. (2000), 'Conservation, regulation, synteny, and introns in a large-scale *C. briggsae–C. elegans* genomic alignment', *Genome Res.*, Vol. 10, pp. 1115–1125.

30. *Bray, N., Dubchak, I. and Pachter, L. (2003),

'AVID: A global alignment program', *Genome Res.*, Vol. 13, pp. 97–102.

31. ★★Brudno, M., Do, C. B., Cooper, G. M. *et al.* (2003), 'LAGAN and Multi-LAGAN: Efficient tools for large-scale multiple alignment of genomic DNA', *Genome Res.*, Vol. 13, pp. 721–731.

32. Nadeau, J. H. and Taylor, B. A. (1984), 'Lengths of chromosomal segments conserved since the divergence of man and mouse', *Proc. Natl Acad. Sci.*, Vol. 81, pp. 814–818.

33. ★★Pevzner, P. A. and Tesler, G. (2003), 'Genome rearrangements in mammalian evolution: Lessons from human and mouse genomic sequences', *Genome Res.*, Vol. 13, pp. 13–26.

34. ★Couronne, O., Poliakov, A., Bray, N. *et al.* (2003), 'Strategies and tools for whole-genome alignments', *Genome Res.*, Vol. 13, pp. 73–80.

35. ★★Gibbs, R. A., Weinstock, G. M., Metzker, M. L. *et al.* (2004), 'Genome sequence of the Brown Norway rat yields insights into mammalian genome evolution', *Nature*, Vol. 428, pp. 493–521.

36. ★★Bourque, G., Pevzner, P. A. and Tesler, G. (2004), 'Reconstructing the genomic architecture of ancestral mammals: Lessons from human, mouse, and rat genomes', *Genome Res.*, Vol. 14, pp. 507–516.

37. ★Kalafus, K. J., Jackson, A. R. and Milosavljevic, A. (2004), 'Pash: Efficient genome-scale sequence anchoring by positional hashing', *Genome Res.*, Vol. 14, pp. 672–678.

38. ★Brudno, M., Poliakov, A., Salamov, A. *et al.* (2004), 'Automated whole-genome multiple alignment of rat, mouse, and human', *Genome Res.*, Vol. 14, pp. 685–692.

39. ★★Brudno, M., Malde, S., Poliakov, A. *et al.* (2003), 'Glocal alignment: Finding rearrangements during alignment', Special Issue on the Proceedings of the ISMB 2003, *Bioinformatics*, Vol. 19, pp. 54i–62i.

40. Eppstein, D., Galil, Z., Giancarlo, R. and Italiano, G. F. (1992), 'Sparse dynamic programming I: Linear cost functions', *J. ACM*, Vol. 39, pp. 546–567.

41. ★Sundararajan, M., Brudno, M., Small, K. *et al.* (2004), 'Chaining algorithms for alignment of draft sequence', in 'Proceedings of the 4th Workshop on Algorithms in Bioinformatics (WABI)' 14th–17th September, Bergen, Norway.

42. ★★Kent, W. J., Baertsch, R., Angie Hinrichs, A. *et al.* (2003), 'Evolutions cauldron: Duplication, deletion, and rearrangement in the mouse and human genomes', *Proc. Natl Acad. Sci. USA*, Vol. 20, pp. 11484–11489.

43. Simon, A., Stone, E. A. and Sidow, A. (2002), 'Inference of functional regions in proteins by quantification of evolutionary constraints', *Proc. Natl Acad. Sci. USA*, Vol. 99, pp. 2912–2917.

44. ★★Thomas, J. W., Touchman, J. W., Blakesley, R. W. *et al.*, (2003), 'Comparative analyses of multi-species sequences from targeted genomic regions', *Nature*, Vol. 424, pp. 788–793.

45. Göttgens, B., Barton, L. M., Chapman, M. A. (2002), 'Transcriptional regulation of the stem cell leukemia gene (SCL) – comparative analysis of five vertebrate SCL loci', *Genome Res.*, Vol. 12, pp. 749–759.

46. ★Cooper, G. M. and Sidow, A. (2003), 'Genomic regulatory regions: Insights from comparative sequence analysis', *Curr. Opin. Genetics Develop.*, Vol. 13, pp. 604–610.

47. ★★Cooper, G. M., Brudno, M., Stone, E. A. *et al.* (2004), 'Characterization of evolutionary rates and constraints in three mammalian genomes', *Genome Res.*, Vol. 14, pp. 539–548.

48. ★★Margulies, E. H., Blanchette, M., NISC Comparative Sequencing Program, *et al.* (2003), 'Identification and characterization of multi-species conserved sequences', *Genome Res.*, Vol. 13, pp. 2507–2518.

49. Wang, L. and Jiang, T. (1994), 'On the complexity of multiple sequence alignment', *J. Comput. Biol.*, Vol. 1, pp. 337–348.

50. Waterman, M. S. and Perlwitz, M. D. (1984), 'Line geometries for sequence comparisons', *Bull. Math. Biol.*, Vol. 46, pp. 567–577.

51. Taylor, W. (1987), 'Multiple sequence alignment by a pairwise algorithm', *CABIOS*, Vol. 3, pp. 81–87.

52. Higgins, D. and Sharpe, P. (1988), 'CLUSTAL: A package for performing multiple sequence alignment on a microcomputer', *Gene*, Vol. 73, pp. 237–244.

53. Thompson, J. D., Higgins, D. G. and Gibson, T. J. (1994), 'CLUSTAL W: Improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice', *Nucleic Acids Res.*, Vol. 22, pp. 4673–4680.

54. Barton, G. J. and Sternberg, M. J. E. (1987), 'A strategy for the rapid multiple alignment of protein sequences', *J. Mol. Biol.*, Vol. 198, pp. 327–337.

55. Taylor, W. (1988), 'A flexible method to align large numbers of biological sequences', *J. Mol. Evol.*, Vol. 28, pp. 161–169.

56. Gotoh, O. (1996), 'Significant improvement in accuracy of multiple protein sequence alignments by iterative refinement as assessed by reference to structural alignments', *J. Mol. Biol.*, Vol. 264, pp. 823–838.

57. ★★Notredame, C., Higgins, D. G. and Heringa, J. (2000), 'T-Coffee: A novel method

for fast and accurate multiple sequence alignment', *J. Mol. Biol.*, Vol. 302, pp. 205–217.

58. ⋆Katoh, K., Misasa, K., Kuma, K. and Miyata, T. (2002), 'MAFFT: A novel method for rapid multiple sequence alignment based on fast Fourier transform', *Nucleic Acids Res.*, Vol. 30(14), pp. 3059–3066.

59. ⋆⋆Edgar, R. C. (2004), 'MUSCLE: Multiple sequence alignment with high accuracy and high throughput', *Nucleic Acids Res.*, Vol. 32(5), pp. 1792–1797.

60. ⋆Van Walle, I., Lasters, I. and Wyns, L. (2004), 'Align-m – a new algorithm for multiple alignment of highly divergent sequences', *Bioinformatics*, Vol. 20, pp. 1428–1435.

61. ⋆⋆Do, C. B., Brudno, M. and Batzoglou, S. (2004), 'ProbCons: Probabilistic consistency-based multiple alignment of amino acid sequences', *Genome Research*, February 2004, (in press) extended abstract in 'Abstract in the Nineteenth National Conference on Artificial Intelligence AAAI', 25th–29th July, San Jose, CA, p. 703.

62. ⋆⋆Bray, N. and Pachter, L. (2004), 'MAVID: Constrained ancestral alignment of multiple sequences', *Genome Res.*, Vol. 13, pp. 693–699.

63. ⋆⋆Blanchette, M., Kent, J. W., Riemer, C. *et al.* (2004), 'Aligning multiple genomic sequences with the threaded blockset aligner', *Genome Res.*, Vol. 14, pp. 708–715.

64. Carillo, H. and Lipman, D. (1988), 'The multiple sequence alignment problem in biology', *SIAM J. Appl. Math.*, Vol. 48, pp. 1073–1082.

65. Altschul, S. and Lipman, D. (1989), 'Trees, stars, and multiple sequence alignment', *SIAM J. Appl. Math.*, Vol. 49, pp. 197–209.

66. Anson, E. and Myers, E. (1997), 'ReAligner: A program for refining DNA sequence multi-alignments', in 'Proceedings of the First Annual International Conference on Computational Molecular Biology', ACM Press, New York, pp. 9–16.

67. Morgenstern, B., French, K., Dress A. and Werner, T. (1998), 'DIALIGN: Finding local similarities by multiple sequence alignment', *Bioinformatics*, Vol. 14, pp. 290–294.

68. Edgar, R. C., Sjolander, K. A. (2004), 'Comparison of scoring functions for protein sequence profile alignment', *Bioinformatics*, Vol. 20(8), pp. 1301–1308.

69. Gotoh, O. (1990), 'Consistency of optimal sequence alignments', *Bull. Math. Biol.*, Vol. 52, pp. 509–525.

70. Vingron, M. and Argos, P. (1991), 'Motif recognition and alignment for many sequences by comparison of dot matrices', *J. Math. Biol.*, Vol. 218, pp. 34–43.

71. Vingron, M. and Argos, P. (1989), 'A fast and sensitive multiple sequence alignment algorithm', *Comput. Appl. Biosci.*, Vol. 5(2), pp. 115–121.

72. Huang, X. and Miller, W. (1991), 'A time-efficient, linear space local similarity algorithm', *Adv. Appl. Math.*, Vol. 12, pp. 337–357.

73. Holmes, I. and Durbin, R. (1998), 'Dynamic programming alignment accuracy', *J. Comput. Biol.*, Vol. 5(3), pp. 493–504.

74. ⋆Ma, B., Wang, Z. and Zhang, K. (2003), 'Alignment between two multiple alignments', in 'Proceedings of the 14th Symposium on Combinatorial Pattern Matching', Lecture Notes in Computer Science Vol. 2676, Springer, Berlin, pp. 254–265.

75. ⋆Kececioglu, J. and Starrett, D. (2004), 'Aligning alignments exactly', in 'Proceedings of the Eighth Annual International Conference on Research in Computational Molecular Biology (RECOMB)', ACM Press, New York, pp. 85–96.

76. ⋆⋆Lee, C., Grasso, C. and Sharlow, M. F. (2002), 'Multiple sequence alignment using partial order graphs', *Bioinformatics*, Vol. 18, pp. 452–464.

77. ⋆⋆Raphael, B., Zhi, D., Tang, H. and Pevzner, P. (2004), 'A novel method for multiple alignment of sequences with repeated and shuffled elements', *Genome Res.*. Vol. 14, pp. 2336–2346.

78. Blanchette, M., Green, E. D., Miller, W., Haussler, D. (2004), 'Reconstructing large regions of an ancestral mammalian genome in silico', *Genome Research*, Vol. 14, pp. 2412–2423.

79. ⋆⋆Miller, W. (2000), 'Comparison of genomic sequences: Solved and unsolved problems', *Bioinformatics*, Vol. 17, pp. 391–397.