



Multiple sequence alignment using partial order graphs

Christopher Lee^{1,*}, Catherine Grasso^{1,2} and Mark F. Sharlow¹

¹Department of Chemistry and Biochemistry, University of California, Los Angeles, Los Angeles, CA 90095-1570, USA

Received on August 3, 2001; revised on September 26, 2001; accepted on October 9, 2001

ABSTRACT

Motivation: Progressive Multiple Sequence Alignment (MSA) methods depend on reducing an MSA to a linear profile for each alignment step. However, this leads to loss of information needed for accurate alignment, and gap scoring artifacts.

Results: We present a graph representation of an MSA that can itself be aligned directly by pairwise dynamic programming, eliminating the need to reduce the MSA to a profile. This enables our algorithm (Partial Order Alignment (POA)) to guarantee that the optimal alignment of each new sequence versus each sequence in the MSA will be considered. Moreover, this algorithm introduces a new edit operator, *homologous recombination*, important for multidomain sequences. The algorithm has improved speed (linear time complexity) over existing MSA algorithms, enabling construction of massive and complex alignments (e.g. an alignment of 5000 sequences in 4 h on a Pentium II). We demonstrate the utility of this algorithm on a family of multidomain SH2 proteins, and on EST assemblies containing alternative splicing and polymorphism.

Availability: The partial order alignment program POA is available at <http://www.bioinformatics.ucla.edu/poa>.

Contact: leec@mbi.ucla.edu

INTRODUCTION

Multiple Sequence Alignment (MSA) is one of the most important tools in bioinformatics, and has a long history of study in computer science and bioinformatics (Needleman and Wunsch, 1970; Smith and Waterman, 1981; Taylor, 1986; Barton and Sternberg, 1987; Higgins and Sharp, 1988; Altschul, 1989; Lipman *et al.*, 1989; Subbiah and Harrison, 1989; Gotoh, 1993, 1996; Thompson *et al.*, 1994; Notredame *et al.*, 2000; Gordon *et al.*, 2001). It continues to be an active field of research, because it poses computational challenges that are only partially solved by

existing algorithms. For pairwise sequence alignment (of just two sequences), a globally optimal solution can be found in $O(L^2)$ time by dynamic programming, where L is the length of the sequences. This algorithm can be extended to align N sequences optimally, but requires $O(L^N)$ time. Whereas the pairwise alignment time of $O(L^2)$ is acceptable for typical biological sequences of genes and proteins ($L < 10\,000$), the exponential time required for aligning larger numbers of sequences by dynamic programming is impractical. Thus optimal alignment of pairs of sequences is performed routinely, but optimal alignment of multiple sequences is generally not possible.

Instead, a variety of heuristic MSA algorithms have been developed, nearly all of them based on progressive application of pairwise sequence alignment to build up alignments of larger numbers of sequences as proposed by Feng and Doolittle (1987). An excellent example of the progressive alignment approach is CLUSTAL, initially released by Higgins *et al.* in 1988 (Higgins and Sharp, 1988). This algorithm builds a MSA through a series of pairwise alignments. Initially, all of the sequences are aligned pairwise resulting in $N(N - 1)/2$ alignments. The scores of these alignments are then used to construct a binary tree of their evolutionary relationships. Finally, the algorithm builds a MSA in the order dictated by the evolutionary tree: the most recently diverged sequences are aligned first resulting in $N/2$ alignment profiles; the $N/2$ alignment profiles are aligned to each other resulting in $N/4$ alignment profiles; and so forth, until all of the sequences have been aligned, resulting in a single alignment profile of the MSA of all N sequences. Finding the scores and constructing the binary evolutionary tree is $O(NL^2)$. Using the binary tree to build the MSA is $O(L^2 \log N)$. Consequently, the overall algorithm runs in polynomial time and can be used to align over a hundred sequences in a reasonable amount of time. As pointed out by Gibson and co-workers (Thompson *et al.*, 1994), this algorithm is greedy by nature and may find a local minimum either because the guide tree is not correct or because alignment errors that happen early on

*To whom correspondence should be addressed.

²Current address: center for Applied Mathematics, 657 Rhodes Hall, Cornell University, Ithaca NY 14853, USA.

in the process of building the MSA get locked in. As a consequence, this algorithm does not guarantee an optimal MSA. Many MSA methods have been developed with different advantages and disadvantages (Taylor, 1986; Barton and Sternberg, 1987; Altschul, 1989; Lipman *et al.*, 1989; Subbiah and Harrison, 1989; Gotoh, 1993, 1996; Thompson *et al.*, 1994; Notredame *et al.*, 2000; Gordon *et al.*, 2001), but globally optimal alignment of larger numbers of sequences remains impractical.

Gibson and co-workers (Thompson *et al.*, 1994) highlighted two major problems with the progressive MSA approach: the local minimum problem, mentioned above, and the choice of appropriate alignment parameters. Their solution was to weight sequences according to their degree of similarity, and to vary gap penalties and substitution matrices during not only the dynamic programming but also the course of building up the full alignment. This strategy was implemented in CLUSTALW, which has been one of the most commonly used MSA programs since 1994 (Thompson *et al.*, 1994).

While we agree with this analysis, we believe there are additional problems in progressive alignment, whose resolution can yield further improvements. Specifically, we view handling of gaps and insertions as one of the most troublesome and least rigorously formulated aspects of MSA. We wish to emphasize that this is not a criticism of any existing MSA method, but rather a way of highlighting opportunities for fundamental rethinking of the problem. We will use the term ‘progressive alignment’ to designate the general class of MSA build-up algorithms, and our usages of this term should not be interpreted as references to (or criticisms of) any existing algorithm. Based on analysis of fundamental problems in gap/insertion representation in progressive alignment, we will present a novel approach based on graph theory that could have broad applicability in MSA and sequence analysis algorithms. Indeed, this approach complements existing MSA algorithms, and can be combined with them in many useful ways, since each method has specific advantages.

PROGRESSIVE ALIGNMENT AND THE MSA REPRESENTATION PROBLEM

To identify new areas of potential improvement in progressive alignment, it is important to define its components clearly. Progressive alignment requires aligning pairs of MSAs, to build up larger MSAs. In practice, however, pairwise dynamic programming is not applied directly to align the pairs of MSAs. Instead, progressive alignment has relied on reducing each MSA to a one-dimensional (1D) sequence which can be used in pairwise dynamic programming sequence alignment. This 1D sequence is a *consensus sequence* or *profile* of the MSA that gives position-specific residue weighting and scoring.

It is assumed that the two cluster MSAs should be aligned exactly as their 1D profiles align, which makes sense if no information about each MSA is lost when it is reduced to a profile. From this point of view, gaps/insertions pose a special challenge (Altschul, 1989; Gotoh, 1996), since they raise the question of what should be included in the profile, and the threat that important information will be lost in the reduction process.

Unfortunately, this reduction of an MSA to a 1D profile inevitably involves loss of information. That is, while the MSA contains all the information to produce the profile, the profile does not contain all the information needed to reconstruct the original MSA. This is demonstrated by the fact that many different MSAs (e.g. with letters in a given column shuffled) would reduce to the same profile. Because a given profile does not map uniquely to a single MSA, it cannot be used to reconstruct the original MSA. This lack of *uniqueness* can also be described as *degeneracy* in the MSA \rightarrow profile mapping.

What kinds of information are lost? Let’s assume that all columns of the MSA are included in the 1D profile (if not, that would immediately constitute a loss of information). While the profile may keep residue and gap frequencies for each column, it has no information on which sequence a given letter comes from. This makes scoring of gaps/insertions especially problematic. Consider the simple MSA below. The profile will contain *all* columns from *all* sequences in the MSA, even though it is likely that no sequence in the MSA (or nature) actually contains all these columns. Thus, any sequence not containing *all* these inserts will now be charged artifactual gap penalties even if its ‘gaps’ are exactly the same as those in one or more sequences in the alignment. Conversely, the only sequence that won’t be charged a gap penalty is the completely artificial sequence constructed by pasting together all columns from all sequences. Alignment of multidomain proteins, for example, would give rise to a gap of hundreds of residues wherever an extra domain was present or missing in one or more sequences, resulting in very large gap penalties. Although one may use position-specific gap penalties to try to diminish this problem (for example, reducing the gap penalty at a position if many sequences are gapped there), this just reveals more problems. For example, if only one sequence in the profile contained an extra domain, we might reduce the gap penalty greatly for those positions, to allow for sequences that lack this domain. However, very low gap penalties are extremely problematic because they cause an exponential increase in the number of higher scoring alignments, even for alignment of a random sequence. And if a new sequence happened to align to this extra domain, it would be wrong to use an extremely low gap penalty for its insertions/deletions within that domain. A 1D profile has no way to correct for this, because it does not know

that these positions are all from the same source sequence.

```

. . . . . ACATGTCGAT . . . . . AGGTG
TGCAC . . . . . TCGATACATAAGGTG
          *           ^

```

Furthermore, consider the two columns marked above. Technically, there is a gap in both columns. One position (^) is a true gap, i.e. a position that one sequence is missing, bracketed on both sides by positions where both sequences align. However, the other position (*) is not actually a gap in the alignment, because both sequences do not even begin to align until several residues further to the right. Aligning the sequence $S = \text{TGCACTCGAT} \dots$ to a profile of this MSA would be charged a 5 residue gap penalty because S lacks ACATG. This gap penalty is purely an artifact of the 1D representation: it vanishes if we reorder the profile by swapping the residues 1–5 with residues 6–10. Now $\text{TGCACTCGAT} \dots$ matches without a gap. This reordering of the consensus does not change the content of the original MSA, since the first five residues of both sequences were not aligned at all.

This reveals that there is tremendous degeneracy in the representation of gaps/insertions within this standard MSA format, which we will refer to as the tabular *Row–Column MSA* (RC-MSA) format. For example, there are $\binom{5+5}{5} = \frac{10!}{5!5!}$ possible orderings of the first ten columns of this MSA that differ only in the order in which these unaligned residues are given. These different RC-MSAs are all equivalent to the original alignment, but will each give rise to *different* gap penalties. There is no ‘right choice’ of which to use; all of these gap penalties are artificial.

We can develop a new approach to MSA based on these considerations. We desire a new MSA representation that eliminates both the degeneracy of the RC-MSA (which causes scoring artifacts), and the degeneracy of the 1D profile (which causes information loss). To be useful for progressive alignment, this MSA representation: (1) should itself be alignable by pairwise dynamic programming alignment, just as a 1D sequence or profile can be; and (2) should not lose any information from the MSA, nor introduce any degeneracy. This raises the interesting question, what is the real content of an MSA? We consider this to be just two pieces of information: *what sequence positions are aligned to each other*; and *the ordering of these positions within the sequences themselves*. We have constructed a data structure that purifies the idea of an MSA down to just these two properties, which satisfies all of the above criteria, and therefore has very interesting benefits for progressive MSA.

ALGORITHM

Partial Order Multiple Sequence Alignment (PO-MSA) data structure

Consider the simple alignment of two sequences depicted in RC-MSA format in Figure 1a. We can instead represent it as a partially ordered graph in which individual sequence letters are represented by nodes, and directed edges are drawn between consecutive letters in each sequence (Figure 1d). In this PO-MSA format, a single sequence is simply a linear series of nodes each connected by a single incoming edge and a single outgoing edge (Figure 1b). Reconstructing the RC-MSA in Figure 1a as a PO-MSA requires two steps. First, the two aligned sequences are redrawn in PO-MSA format, with dashed circles indicating aligned letters (Figure 1c). Second, the letters that are aligned and identical are fused as a single node, while the letters that are aligned but not identical are represented as separate nodes that are recorded as being aligned to each other (Figure 1d). When letters are fused as one node, the resulting node stores information about all of the individual sequence letters from which it was derived, specifically the ID of the original sequence(s) and the index of the letter’s position within that sequence. Thus it is possible to trace the path of each individual sequence through the PO-MSA. The PO-MSA format contains all the information of a traditional MSA, but represents it as a graph compacted for minimal node and edge counts. Redundant edges are removed, i.e. a given pair of nodes will be connected by at most one edge. A node may have any number of incoming or outgoing edges. These edges simply indicate the preceding or subsequent letters of every sequence that passes through that node.

This representation corresponds exactly to the two components that we identified above as the true content of an MSA. *What sequence positions are aligned* is indicated by node fusion/alignment. The *ordering of these positions within the sequences* is indicated by the directed edges. Since the representation consists of only these two features, which correspond exactly to the two forms of content of an MSA, it represents the MSA without loss of information and without degeneracy. In other words, there is a one to one mapping: each PO-MSA maps to a unique MSA, and each MSA maps to a unique PO-MSA. From the MSA we can construct the PO-MSA, and vice versa.

We use the mathematical term ‘partial order’ for this data structure, because our graphs obey true linear (1D) ordering only in regions of nodes with single outgoing edges. In a linear order (such as the number line of integers), for all distinct nodes i, j it is guaranteed that they are ordered, i.e. either $i < j$ XOR $j < i$, where the ordering relation ‘ $i < j$ ’ is defined to mean ‘there exists a path of one or more directed edges from node i to node j .’ We will refer to this definition as ‘linear’ or 1D ordering.

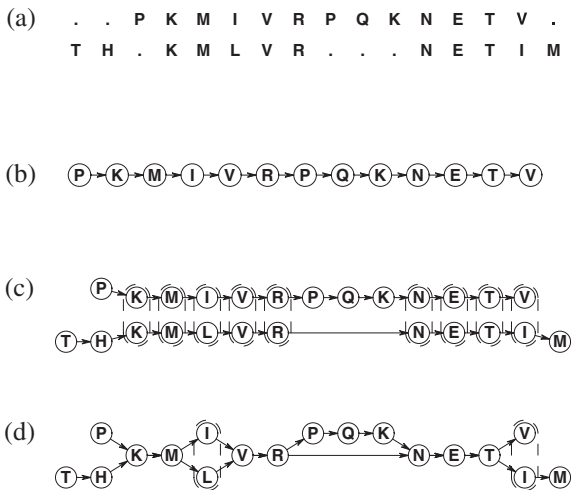


Fig. 1. MSA in the POA representation. (a) RC-MSA representation of a pairwise protein sequence alignment. (b) A single sequence in PO-MSA format. (c) Two protein sequences in PO-MSA format aligned to each other. Dashed circles indicate that two nodes are aligned. (d) PO-MSA representation of a pairwise protein sequence alignment. Dashed circles indicate that two nodes are aligned.

In a partial order graph, by contrast, there may exist nodes i, j such that no path of directed edges exists from i to j or from j to i (i.e. NOT $i < j$ AND NOT $j < i$). When a node n has multiple outgoing or multiple incoming edges e_1, e_2, \dots , we define n as a ‘junction node,’ and the edges e_1, e_2, \dots and the nodes and edges they connect to n as ‘branches.’ There exists no path of directed edges from the nodes on one branch to the nodes on the other branch. Within each branch the nodes are ordered with respect to each other, but the nodes of one branch have no ordering relation to the nodes on the other branch. Our PO-MSA data structure belongs to the well-studied class of data structures known as Directed Acyclic Graphs (DAGs).

Figure 2 shows a PO-MSA compared with the conventional RC-MSA representation of a table of rows (sequences) and columns (aligned letters). Gaps in the RC-MSA appear in the PO-MSA as edges that jump one or more nodes. The data compression property of the PO-MSA is readily apparent. It reduces the RC-MSA to a minimum description consisting of a consensus sequence plus the deviations of individual sequences. The partial order graph structure is also equivalent to the representation of an alignment of N sequences as a path through an N -dimensional MSA matrix. Figure 3 shows the N -dimensional representation versus the partial order graph representation of a simple alignment of three sequences.

Dynamic programming Partial Order Alignment (POA)

Standard dynamic programming sequence alignment (Needleman and Wunsch, 1970; Smith and Waterman, 1981) can be extended to work with partial orders. In this paper we present a simple algorithm for aligning a linear sequence to a PO-MSA (representing many aligned sequences, or possibly a single sequence).

The extension can be illustrated in the following way (Figure 4). Standard dynamic programming alignment of two linear sequences can be represented as a two dimensional (2D) matrix, whose two axes correspond to the two sequences. A given point (n, m) in the matrix corresponds to a pair of sequence positions (n in sequence 1, m in sequence 2). For a given pair-position, three basic moves are possible: a diagonal ‘alignment’ move indicating that n and m are aligned; and horizontal and vertical moves indicating, respectively, n as an insertion relative to m , or m as an insertion relative to n . The set of all possible paths across the 2D matrix constructed from these moves represents all possible alignments of the two sequences allowing the ‘edit operators’ of identity/substitution (diagonal move) and insertion/deletion (horizontal and vertical moves).

POA extends the dynamic programming method of Needleman–Wunsch in a natural way. We replace one of the linear sequences by a partial order containing branching. We transfer the partial order structure to the 2D matrix by grafting a copy of its bifurcations across the matrix, forming additional dynamic programming matrix surfaces that are joined exactly as the individual branches are joined in the PO-MSA (Figure 4b). Thus, the 1D branches of the partial order become bifurcated surfaces in the partial order matrix. Now we must extend the set of possible moves appropriately. On a given surface, the POA behaves the same as the standard 2D alignment, and the same set of three moves (diagonal, horizontal, vertical) are allowed. At junctions where multiple surfaces fuse, we extend the horizontal and diagonal moves to allow them to go onto *any* of the incoming surfaces that meet at this junction. Thus for the simplest case where two branches join, the allowed moves are: two diagonal moves (one from each incoming surface), two horizontal moves (again, one from each incoming surface), only one vertical move, and a ‘start’ move (explained below in Section **Alignment scoring**).

Given the partial order move set, dynamic programming sequence alignment (Needleman and Wunsch, 1970; Smith and Waterman, 1981) can now be applied in the usual way. At a given cell (n, m) in the matrix, the scores for all possible moves at this point are calculated, and the move with the maximum score is selected and saved at this cell. The score for each possible move is just the sum of the transition score associated with that move (for a

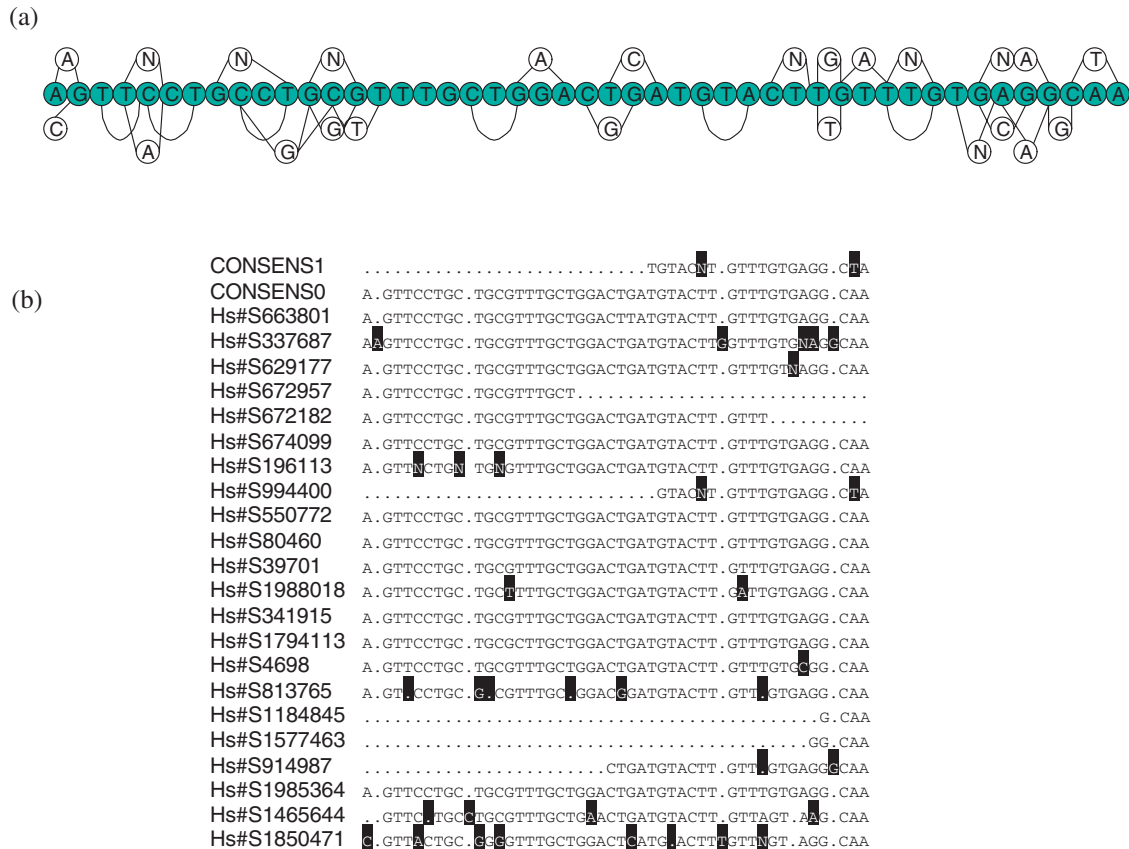


Fig. 2. Partial order alignment representation of UniGene EST sequences. Part of an alignment generated by POA of EST sequences from UniGene cluster Hs. 100194: (a) PO-MSA representation; (b) traditional RC-MSA representation of the same data. Only part of the very large alignment is shown.

diagonal move, the substitution score $s(n, m)$ for aligning residue n to residue m ; for a horizontal or vertical, the gap penalty Δ , plus the saved score $S(p, m - 1)$ at the predecessor cell pointed to by the move.

$$S(n, m) = \max \begin{cases} S(p, m - 1) + s(n, m) \\ S(p, m) + \Delta(m) \\ S(n, m - 1) + \Delta(n), \end{cases} \quad (1)$$

considering all predecessor nodes p that have a directed edge from $p \rightarrow n$.

The only difference from standard sequence alignment, in which there can be only one predecessor node, is that there may be many predecessor nodes p for a given letter n in the partial order.

Alignment scoring

In this paper we used Smith–Waterman dynamic programming sequence alignment (Smith and Waterman, 1981), which seeks only to align those regions that have a positive alignment score. Smith–Waterman differs from so-called ‘global alignment’ in that it does not charge a gap

penalty for unaligned ends of sequences, whereas global alignment does. This is implemented by allowing an additional ‘start’ move at each cell, whose total score is just zero, and corresponds to asserting that none of the preceding residues in either sequence are aligned. It should be emphasized the POA could in principle be used with any of the common variations of dynamic programming alignment: global, Smith–Waterman, ‘overlap,’ or ‘repeat.’

Scores are calculated for each cell in order, starting from the origin (0, 0) and filling in the complete partial order matrix, to the end-point (N, M) , where N is the number of nodes in the PO-MSA G , and M is the length of the new linear sequence S being aligned. The cell with the highest score is taken as the end of the best alignment, which is traced backwards through the full alignment path, by simply following the saved best-move from that cell to its predecessors, iteratively, until the path is terminated by a ‘start’ move representing the beginning of the alignment. This produces a set of aligned letter/node pairs (i_S, j_G) from the sequence and the PO-MSA (e.g. Figure 1c).

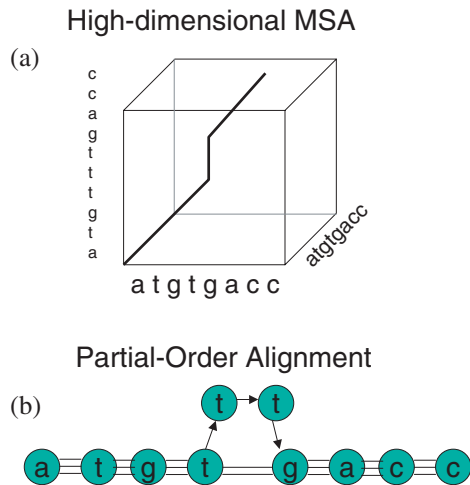


Fig. 3. POA maps the high-dimensional MSA path to a reduced graph representation. (a) MSA can be represented as a path through a high-dimensional space in which the dimensions of the space correspond directly to the individual sequences (and thus have a length equal to the length of the corresponding sequence). (b) POA translates the high-dimensional MSA path into a graph representation in which each node corresponds to a set of aligned letters in the alignment.

Because of gaps, a given sequence letter l_S may not be aligned to any node in the PO-MSA at all; that is, it is not included in the (i_S, j_G) alignment pair mapping.

Construction of the partial order graph

The sequence S is first converted into a trivial PO-MSA (e.g. Figure 1b) with each letter stored in a separate node and with a single directed edge between each successive node. The sequence ID and residue position index of each letter are stored on each node. This trivial PO-MSA is joined to the original PO-MSA G through a process of node fusion. When two nodes are fused, the lists of sequence IDs and residue position indices stored on the individual nodes are combined as a single list stored on the fused node. While this sequence information is ignored by the simple POA alignment algorithm described above (in order to enable recombination during MSA; see Section **Discussion**), it is essential for analysis of the resulting PO-MSA. We will describe in a separate publication a series of algorithms we have developed for analyzing PO-MSAs.

PO-MSA nodes are fused according to the (i_S, j_G) mapping as follows. If node i_S and node j_G are aligned and have identical letters then they are fused (e.g. letters NET at the right of Figures 1c, d). If node i_S and node j_G are aligned, but do not have identical letters, and node j_G is already aligned to a node k_G in G whose letter is identical to node i_S , then node i_S and node k_G are fused.

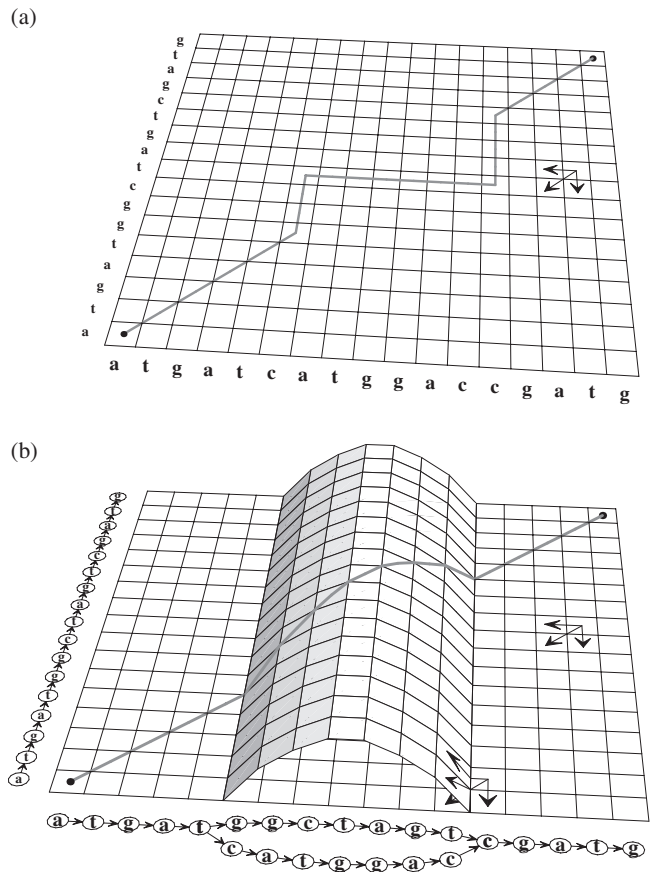


Fig. 4. Dynamic programming matrix for POA algorithm. (a) Dynamic programming matrix for Needleman–Wunsch sequence alignment algorithm. The optimal global alignment path is shown. (b) Dynamic programming matrix for POA algorithm. The optimal global alignment path for aligning a single sequence represented as a DAG to two sequences represented as a PO-MSA is shown. The matrix bifurcates in the middle since the single sequence DAG can align to either sequence stored in the PO-MSA. At the bifurcation point five moves are allowed since the sequence can align to either the first sequence traversing the upper dynamic matrix, or the second sequence traversing the lower dynamic matrix. At every other point in the dynamic matrix, not corresponding to a bifurcation point, the usual three moves of insertion, deletion, and match are allowed.

Finally, if node i_S and node j_G are aligned, but do not have identical letters, and node j_G is not already aligned to any node in G with an identical letter to node i_S , then node i_S and node j_G are recorded as being aligned to each other (e.g. letter L of KMLVR in Figures 1c, d). The nodes of PO-MSA S that are unaligned in the (i_S, j_G) mapping (e.g. letters TH at the left of Figures 1c, d) are not altered. Finally, we remove redundant directed edges (i.e. more than one edge connecting the same pair of nodes).

Iterative partial order alignment

In this paper, we have applied POA to MSA in the simplest possible way. To construct a PO-MSA of a set of sequences, they are simply aligned one after another to a growing PO-MSA, using the algorithm described above for aligning a single sequence to a PO-MSA. All the results in this paper were obtained in this manner.

Computational complexity

Since the dynamic programming algorithm can be implemented in the usual way for sequence alignment to a partial order, the only increase in its time complexity over standard sequence alignment arises from the possibility of multiple predecessor nodes p . For n_p predecessor nodes, the number of moves whose score must be calculated in (1) is $2n_p + 1$. For standard sequence alignment, where $n_p = 1$ by definition, there are exactly three moves, and the total time complexity for aligning two sequences of length N , M is $O(3NM)$. For the POA algorithm, the total time complexity of aligning a sequence of length M to an N node PO-MSA with \bar{n}_p average number of predecessors per node is $O((2\bar{n}_p + 1)NM)$. Thus, the time complexity increases only linearly with the average number of branches per node. The space complexity for POA, like standard pairwise dynamic programming sequence alignment, is $O(NM)$, for storing the best score traceback. The time complexity for construction of the (i_S, j_G) mapping by traceback, and for construction of the new partial order graph, is just $O(\min(N, M))$, so these stages are also linear. The space complexity for storing the PO-MSA is $O(N + L)$, where N is the number of nodes in the PO-MSA and L is the total number of letters in all of the sequences aligned in the PO-MSA.

Moreover, the average partial order branch number \bar{n}_p tends to increase only slowly for biologically meaningful alignments, because the partial order representation acts like a data compression algorithm, since it fuses matching sequence letters into a single node, removing redundancy from the representation of the sequence alignment. In many types of alignments there is a high degree of matching sequence. Alignments of transcript sequences such as ESTs, for example, have greater than 90% identity, so \bar{n}_p is only slightly greater than one. In these cases, POA incurs very little extra computational cost, while significantly expanding the capabilities of the alignment algorithm for dealing with complex structures that occur in biological sequences.

IMPLEMENTATION

We have implemented the POA algorithm in the program POA, which is available for download from <http://www.bioinformatics.ucla.edu/poa>. The program was written in C in 1998, and supports a variety of alignment options.

This program can take as input a set of sequences in FASTA format, and output a tabular form of the alignment in the standard PIR or CLUSTAL alignment format, or the PO-MSA representation as a simple text file. It has been used extensively in a high-throughput production environment. It has been essential for our analyses of the human genome (Table 1), such as Single Nucleotide Polymorphism (SNP) discovery (Irizarry *et al.*, 2000) and alternative splicing analysis of human genome data (Modrek *et al.*, 2001).

Partial order structure in biological sequence alignments

Multidomain proteins. The utility and interest of POA can easily be seen by applying this technique to MSA of mammalian proteins. As an example, we applied the simple iterative POA algorithm to several human protein sequences containing SH2 domains: ABL1, CRKL, GRB2, and MATK, obtained from SwissProt (Figure 5). Using the program POA and the BLOSUM80 scoring matrix, we aligned ABL1 and CRKL to create a partial order graph. Next, GRB2 and MATK were aligned to this partial order graph. The resulting alignment clearly shows the multidomain structure and relationships of these proteins. At the center of the graph, all four sequences merge, indicating that they share a common domain. The SwissProt annotation records indicate that this aligned region corresponds to an SH2 domain found in all four sequences. To the left of this region, three of the sequences (MATK, ABL1, GRB2) are aligned for about 40–50 amino acids. According to the SwissProt annotations, this corresponds to an SH3 domain found in these three sequences. To the right side of the shared SH2 domain, MATK and ABL1 split off from the other two sequences, and are aligned for about 235 amino acids, a kinase domain. MATK and ABL1 then diverge at their C-terminal ends. After the SH2, GRB2 and CRKL are aligned for 60 amino acids, an SH3 domain, at which point GRB2 ends and CRKL continues for another 118 amino acids.

The full details of this alignment are shown in Figure 6, in the conventional alignment table format. This format obscures the partial order structure of the alignment, but is convenient for showing the sequence letter matches. POA does a good job of aligning the sequences, as judged by the superposition of the annotated domains (shown by black or gray shading). Despite the low level of identities (shown in bold), particularly in the first SH3 and SH2 domains, the domains appear to be aligned well. The most visible question is the end residues of domains, which are sometimes left unaligned, e.g. the first 8 residues of MATK's SH3 domain. This is an issue of the BLOSUM80 scoring matrix, not an issue of the POA algorithm. The alignment that was chosen has a better BLOSUM80 score

Table 1. Partial order alignment of all human expressed sequence data

Total EST clusters aligned (UniGene Feb 2001)	89 341 clusters
Total number of EST and mRNA sequences aligned	2 357 605 sequences
Total nucleotide length of EST and mRNA sequences aligned	1 045 666 301 nt
Largest alignment: Hs. 111334	4 704 sequences
Largest alignment: total nucleotide length of sequences aligned	2 201 252 nt
Largest alignment: longest individual sequence	2 095 nt
Largest alignment: average sequence length	468 nt

Using the program POA we aligned over 2 million human ESTs in UniGene clusters (Irizarry *et al.*, 2000, 2001; Modrek *et al.*, 2001). These PO-MSAs were the basis of our genome-wide analysis of SNPs and alternative splicing, and were checked both by extensive visual analysis, and by experimental testing of SNPs identified using these alignments (Irizarry *et al.*, 2000).

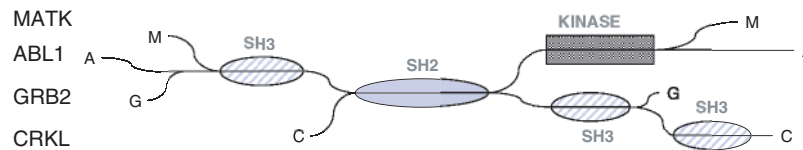


Fig. 5. Partial order alignment of four human SH2 domain containing proteins. Four sequences (SwissProt identifiers MATK_HUMAN, ABL1_HUMAN, GRB2_HUMAN, CRKL_HUMAN) were iteratively aligned by POA to produce this PO-MSA. The alignment is shown as a schematic of the partial order structure (see Section **Algorithm**) indicating where the sequences align (lines that join) and diverge (lines that split apart). The termini of each sequence are labeled with their respective first letters. The protein domains annotated in the SwissProt records for these proteins are shown as ovals and rectangles. Full details (the individual letter nodes and edges, including minor branchings) have been suppressed to make the overall structure clear. The full details of this PO-MSA are shown in Figure 6.

than aligning these eight residues to the ABL1/GRB2 SH3 residues. Similarly, the last 13 residues of GRB2's SH2 domain (TSVSRNQQIFLRD) are not aligned to the other sequences' last SH2 residues. Again, this is due to the scoring matrix/gap penalty parameter, not the POA algorithm. Since the connection between the SH2 and next SH3 domain is 14 residues shorter in GRB2 than in CRKL, the algorithm must place these gaps somewhere. And the alignment chosen is given a better BLOSUM80 score than fully aligning the last few residues of the SH3 domain (because there is so little homology).

This example shows that real alignments of biological sequences have a partial order structure. The alignment in Figure 5 contains six major branch points, all of which are essential to an accurate depiction of what the four sequences share in common, versus where they diverge. These branching structures are not a trivial reflection of standard ways of classifying sequence alignments, such as subdividing the alignment into a tree of homology families by increasing similarity. For example, GRB2 is more similar to CRKL at the C-terminal end. Together these two sequences diverge from ABL1 and MATK after the SH2 domain. However, at the N-terminal side of the SH2 domain, the situation is reversed. There, GRB2 is more similar to ABL1 and MATK, and all three split away from CRKL. This does not fit the expectation that we might be able to explain these structures in phylogenetic terms (A

is more similar to B than to C or D, therefore A and B will branch away from C and D). Not only is partial order structure (branching) truly present in the alignment, but it expresses a more complex set of relationships than can easily be discovered by a phylogenetic tree.

This also demonstrates that the traditional set of 'edit operators' used in pairwise sequence alignment (substitution, insertion and deletion) is inadequate for multidomain protein MSA. Specifically, a new edit operator for the biological process of *homologous recombination* is needed. Because GRB2 aligns with MATK and ABL1 at its N-terminus, but after the SH2 domain splits away and aligns with CRKL at its C-terminus, GRB2 corresponds to a 'recombination' of MATK/ABL1 and CRKL, with the SH2 domain as the recombination point. GRB2 could not be aligned with these three proteins without including the recombination edit operator. Otherwise it would be forced to make an either/or choice between aligning with ABL1 and MATK at its N-terminus, versus aligning with CRKL at its C-terminus. We will describe how POA handles the recombination edit operator in more detail in the Section **Discussion**.

EST assembly. We have also applied POA to mRNA transcript fragments such as human ESTs (Boguski and Schuler, 1995) (Figures 2, and 7). The partial order structure of EST alignments is biologically meaningful. Even though branch points may be infrequent in EST

abl	..leiclkivgckskkglssssccyleaalqrpvadfdpqqjseaarwnsknllagpse
matkmagrgslsvwrafhgcdsaeeelprvprflrwhpppvs
abl	ndpnl f...VALYDFVASGDN
grb2 ea IAKAYDFKATADD
matk	armptrrwvpgtgcitckehtrpkpgELAFRRKGDVVVIL.EaEenKSNYRVKkhtSGQEG
ablTLSitKGEkLRVLgynhn.geWcEAQtk.NGQ.G
grb2ELSFRRGDILKVLnEeCD.QNWYKAEI..NGKDG
matk	LLA AgALrEr.....EALsaePKlslmp FRK GISGQEAQQQLQ.ppEDGLFLVRESAR
abl	WPSNYItipv.....NSLEKHS.....WYHGPPVSRNAAYLLSgIh.GSFLVRESSE
grb2	F IPKNYI.....eMKRHP.....PFGKlprAAEEMLSKQRHDGAFIRESSES
crklmsaarFDSsDRaA.....WYMGVPSRQEAQLRQQRH.GMFLVRDSST
matk	APGD VVLcVSPGrDVLHYRV.lhrDGHLtIdEaVfFcNLmDNVEHYskdkaIcTkLVRP
abl	SPGGrSISLRYegRVyHYRINTasDGKLYVssEsRFNTLaELVhHSTVadGLiTTLhyF
grb2	APGDFSLVKFGNDVqHFKVlrdgaGK.YFlwvVKFNSLmELVDYHRS.....TSV...
crkl	SPGDVVLsVSeNSRVeHYlINSlpN-RPKlqDqe.FDHLpalLEPYK.ThyldtTLLeF
matk	..KRKHgTksaeelaraagWllnlqhlTLgaqIGeGEFGaVlQGeY..lqqkVAVKNIK
abl	PKR KNPTVYgVS.PNYdkWemertdITMkhkLGGQYGeYVeGvWkysltVAVKTLK
grb2	..sRNG.qIIF..lR..D
crkl	PR YpsPpMgsVSApN
matk	DVt .eoeFlDEEAVMFKQhENLVRLLGVlIHog.LYlVmEHVSQNLVNFIRtrgRaVl
abl	DzmevEeFLkEaAVMKEIKhNPLVOLLGvctReppFYIItEIMTYGNLLDYLReenRgeV
matk	N eaqLlqSPSHVAegMEYLEsKKLVHRDLAARNILVsEDlVaKVSDPGLAK...aErkgI
abl	Navv LlyMaTQISaMEYLEkKNFtIHRDLAARNCLVgENhLwKVADFGLSRLmtgDtyta
matk	IS .SRLPVKWTAPALKhgKfTeKSDVWAFGVLlWbVfSYGArPYKMaLkEvsEaVEK
abl	h agAKFPKWTAPESLaYeKFSiKSDVWAFGVLlWbIaTYGmsPYpIdLaQVYeLEK
matk	YRME pEGCpPpVHvLMasCWEaePA.RPp
abl	YRME rPEGCpEkyVeLMrACWwNPSDRPsFaEilqafetmfGessisdevekeIlgkqv
abl	rgavstllqapelptktrtsrraaehrdtdtvpemphskggesdpldhepavspllprk
abl	ergpppegglinederllpkdkntlfalslkkkkktaptprkrsssfremdgpperrgag
abl	eegrdisnalaftpldtadpaksppksgagvngalresgsgfrshplwkksstlts
abl	srlatgeeeeggssskfrlrsccasvphgakdtewsvtlprldqstgrqfdsstfggh
abl	ksekpalprkragenrdsqvtgrtvtpprllvkneeaaadevfkdimesspgssppnltp
abl	kpIrrgvtvapasgplphkeeaegsalgtpaaeevptpskagsgagpqtkskpaeesvr
abl	<u>rrhkhssespprdkgklrslrkppppppaasagkaggkpsqpsqaeeavlgaktkat</u>
abl	sIvdavnsdaakpsqpegllkppvlpatpkqgsakpsqstpspapvstlpsasallaqd
abl	gsstafipIlistrvslrkrtrqperiasgaikgvvdstealcIairnseqmasha
abl	vleaagnlytfcvsvydsiqmnrnkfafreainklennlrelqicpatagscpaatqdfs
abl	kllsvkeisdivr.....
grb2IeqvqOptVYQALFDfPqEdgE.LgFRGGDFtHVMdNsDpNwW
crklLptaedNleYVRTLYDF.PgnDaEdLpFKGEILVlIEKpEeQW
grb2	<u>kgach</u> .GQTGMFPrnYVtPvNRNv
crkl	sarnkdGRVGMlPvpYVeLkVRS.sphgkhgnrnsnsgyipepahayaqqtltplpavs
crkl	gsppaaItplstqngpvpfakaiqrvcaydktalalevgdivkvtrmningqwegevn
matk PK LaekIareLrsagapasvsgqdadgtsprsqep
crkl	<u>gkgl</u> lfpfthvkifdpqndens

Fig. 6. Partial order alignment of four human SH2 domain containing proteins (details). The PO-MSA from Figure 5 is shown here in a conventional alignment table (RC-MSA) format, in which each column of the table represents a set of aligned residues, with a dot (.) as the gap symbol. Note that the tabular format creates the appearance of many artifactual gaps, which are not present in the PO-MSA, and are not real gaps. Identities are shown in bold, and similar residues (with a BLOSUM80 score ≥ 0 to at least one other aligned residue) are shown in uppercase. Protein domains annotated in the SwissProt records for these proteins are shown as follows: kinase domain (black); SH2 domain (gray); SH3 domains (light gray); proline-rich domain (underlined). The N- and C-terminal residues of each sequence are indicated in light gray. Separate sections of the alignment are separated with a horizontal line; if a sequence is entirely gapped in a given section, it is not shown.

alignments, they are very important features: sequencing errors, polymorphisms, alternative splicing, initiation, or polyadenylation, and paralogous gene sequences. This figure also shows the efficiency of the PO-MSA representation. For sequences with a very high level of identity (such as clustered ESTs) the partial order graph achieves a very high level of compaction through fusion of aligned, identical letters. Such a partial order graph

effectively consists of a ‘consensus’ sequence of nodes, plus occasional ‘polymorphic’ nodes where an individual sequence diverges from the consensus (Figure 2a). A dense number of substitutions, deletions and inserted letters is characteristic of the high level of sequencing error in ESTs. The PO-MSA representation effectively handles this sequencing error, making it easy to find consensus sequences, called heaviest bundles, using graph algorithm methods which we will describe elsewhere. Using POA, the ESTs from UniGene cluster Hs. 1162 were aligned to each other and the HLA-DMB genomic sequence. The resulting POA, shown in Figure 7, has four consensus sequences (heaviest bundles) of nodes, corresponding to four different alternatively spliced mRNA forms of the gene. This alignment was produced by POA as part of our genome-wide study of alternative splicing in human EST sequences (Modrek *et al.*, 2001). This work depended on our ability to easily find consensus mRNA forms along with their associated splice sites in the PO-MSA data structure using simple graph algorithm techniques.

Iterative POA gives an efficient EST assembly algorithm capable of working with very large alignments. For example, in one project analyzing all available human EST data for SNPs (Irizarry *et al.*, 2000) and alternative splicing (Modrek *et al.*, 2001), we constructed approximately 90 000 alignments totaling about 2 million distinct EST sequences (Table 1). However, the sizes of the EST clusters were very heterogeneous. The largest number of EST sequences aligned by POA in a single MSA was about 5000 sequences, in approximately 4 h using an inexpensive Pentium II PC. By contrast, other approaches often impose much more restrictive size limitations. For example, the computationally sophisticated analysis of Burke *et al.* (1998) reports that for the same set of human UniGene clusters they could not process 359 clusters (average size 297 sequences per cluster) because they were too big to be aligned, even using highly optimized MSA software (TIGR_msa) and hardware (MasPar massively parallel computer), both designed specifically for this problem. Based on these data it appears that POA may extend the number of ESTs that can be easily aligned by a factor of 10 or more.

POA is able to scale successfully to such extremely large alignments because of the advantageous computational complexity of this algorithm. POA’s computation time for aligning N homologous EST sequences grows only linearly with the number of sequences. CLUSTAL, by contrast, must construct $O(N^2)$ pairwise alignments (of all possible pairs of sequences), before it even begins MSA. This makes alignment of very large numbers of sequences impractical. For small alignments (e.g. 50 sequences) POA was generally faster or the same speed as CLUSTAL. For larger numbers of sequences it was substantially faster. Thus, partial order sequence

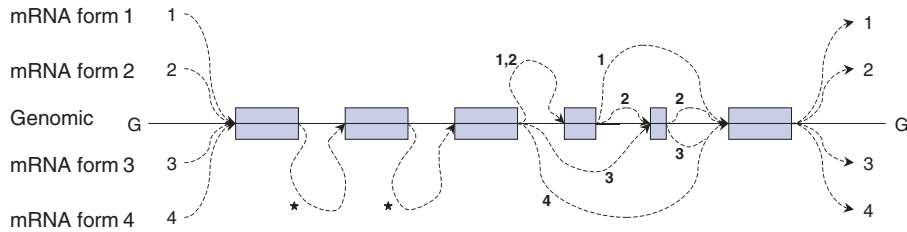


Fig. 7. Partial order alignment of four HLA-DMB mRNA forms and genomic sequence. PO-MSA of the ESTs from UniGene cluster Hs. 1162 aligned to *HLA-DMB* genomic sequence. Consensus sequences (heaviest bundles) corresponding to four alternate mRNA forms are shown. Exons are shown as rectangles. Non-exonic regions of the genomic sequence are shown as a solid line. Dotted lines correspond to the mRNA sequences, and represent their starts, ends, and splices relative to the genomic sequence. Each splice is labeled with the IDs of the mRNA forms containing that splice. The * symbol indicates that all four mRNA sequences share that edge.

alignment offers important advantages even just considering performance and scalability for large data analysis projects.

DISCUSSION

Alignment ordering and pairwise optimal alignment inclusion

To understand the useful properties of the POA algorithm, we must first consider the difficulties that MSA algorithms face. In existing progressive MSA methods, it is important that sequences be aligned in ‘the right order’ (i.e. most similar sequences first), because there is no algorithmic guarantee that the overall alignment process will actually find an optimal alignment for any given sequence (Higgins and Sharp, 1988; Mevissen and Vingron, 1996; Notredame *et al.*, 1998). In particular, if sequences were aligned in a random order, this would produce alignment errors, e.g. two highly similar sequences *S* and *T* might be aligned differently than their optimal pairwise alignment. Why is this? In each progressive alignment step, each cluster MSA must be reduced to a 1D profile, which no longer represents the full complexity of the individual sequences. Thus, when two clusters of sequences are aligned in such an algorithm, there is no guarantee that any pair of sequences from the two clusters will align optimally.

The simple iterative alignment procedure presented in this paper, which aligns sequences in the order in which they are given, will seem very unsophisticated relative to CLUSTAL’s extensive consideration and optimization of the order in which sequences must be aligned. Our alignment method is certainly not immune to order-dependency effects in the MSAs that it constructs. We have observed some differences in large PO-MSAs, particularly in regions of marginal alignment score, depending on the order in which the sequences were added. It is possible to apply standard progressive alignment approaches to ordering the alignment steps (Feng and Doolittle, 1987), by an extension of the algorithm (see Section **Conclusion**). However,

the simple POA algorithm can produce good MSAs of complex (e.g. multidomain) sequence families (e.g. Figures 5 and 6), because the PO-MSA data structure stores all sequences at all stages in the progressive alignment process. When a new sequence *S* is aligned to an existing PO-MSA *G*, POA considers every possible alignment of *S* versus every possible sequence in *G*. There is no possibility of the algorithm ‘missing’ an optimal alignment of *S* against its closest homologous sequence *S'* in *G*, and this is true at every stage in the buildup of the MSA. In this sense, it doesn’t matter greatly whether *S* and *S'* are aligned as the first step of the MSA, or at very different stages of the buildup, because they are guaranteed to find each other at any step.

POA is efficient. It can simultaneously consider all possible alignments of sequence *S* against sequences *T*, *U*, *V*... in *G*, faster than separate pairwise alignments. This is because the partial order graph squeezes out redundancy from the representation of these sequences, by fusing aligned, identical letters. All places where *T*, *U*, *V*... are the same will be represented only once in the graph; additional nodes (requiring additional computation) will only be present where there is new information (i.e. where a sequence diverges from the others). In the worst case, where all the sequences *T*, *U*, *V*... have no homology to each other, the number of nodes in the PO-MSA will simply be the sum of the number of letters in the sequences aligned so far, and the total time complexity for aligning all the sequences increases to $O(N^2L^2)$, where *N* is the number of sequences and *L* is the length of each sequence. Thus, in the worst case scenario POA’s time complexity is equivalent to that of CLUSTAL.

Homologous recombination

A specific sequence alignment algorithm can be considered to be a model for how sequences can change over time. For example, pairwise Needleman–Wunsch align-

ment corresponds to a model that allows two edit operators, *substitution* and *insertion/deletion*. More complex edit operators such as *duplication* (in which an extra copy of part of a sequence is inserted next to it) or *translocation* (in which part of a sequence may be moved to a different location in the sequence) have also been modeled in other algorithms. Any edit operator not explicitly included in a given algorithm, will not be permitted in the alignments produced by that algorithm. Thus, it is important that the algorithm's edit operators match the real-world process by which the sequences it's aligning evolved from a common ancestor.

Our analysis of multidomain protein alignment (Figures 5 and 6) demonstrates that *recombination* is an important edit operator for biological MSAs. In biology, recombination is an important mechanism of gene exchange and generation of genetic diversity. Moreover, most organisms have been shown to contain machinery for *homologous recombination*, that is, crossing over from one genetic sequence to another specifically within regions of homology between them. We can state this recombination edit operator precisely: it means aligning part of a sequence S to one sequence T , and the next part of S to another sequence U , only where T and U are homologous (i.e. aligned, with at least one identity) at the recombination point. For example, if one of the sequences T had domains ACD, and another sequence U had domains BCE, we would need the recombination operator to align the domain sequence ACE to T and U .

We can analyze MSA methods in a new light by posing the following problem: given a set of sequences $T, U, V \dots$ with known alignment relationships, state an efficient algorithm for finding the optimal alignment of a new sequence S to all of them simultaneously, allowing the new edit operator *recombination* in addition to the usual edit operators of *substitution* and *insertion/deletion*. Adding this new edit operator would appear to greatly increase the computational complexity of sequence alignment, since we must now consider all possible recombinations of the set of sequences $T, U, V \dots$ and try aligning S to all of them.

One efficient algorithm for solving this problem is simply POA. Since the partial order graph representation fuses aligned letters from different sequences as a single node in the graph, when a new sequence is being aligned to the partial order, it can switch from aligning to one sequence to aligning to another at these junction nodes. Thus, POA also considers all possible recombinations of $T, U, V \dots$ when aligning S . The traversals in G that represent the sequences $T, U, V \dots$ in G are only a subset of all traversals of G . The remaining traversals represent recombinations in which parts of multiple sequences are linked through junction nodes where two or more sequences are aligned and identical. Since POA aligns S

against all possible traversals of G , it considers all possible recombinations that pass through junction nodes.

If the number of branches in the alignment is b , the number of possible recombinations grows exponentially (as $O(2^b)$ if we assume no two branches are at the same node). The time complexity of POA, by contrast, grows at only $O(2\bar{n}_p + 1) = O(2(1 + \frac{b}{N}) + 1)$, where \bar{n}_p is the average number of branches per node, and N is the total number of nodes in the graph. We would have to add many branches to the graph before we would notice even a slight increase in the computation time. POA is able to consider recombinations efficiently because the graph structure strictly limits the set of all possible recombinations of the sequences, to just those that pass through regions of homology (i.e. junction nodes where two or more sequences are aligned and identical).

CONCLUSION

Should MSAs be linear?

The examples in this paper suggest that it may be productive to question basic assumptions about sequence alignment that underlie most work in the field. A single sequence is by definition a linear object. Thus it is natural to assume by extension that an alignment of sequences would also be linear. However, a MSA is not necessarily a linear object, but rather a *partial order*, because it joins together multiple sequences that can diverge from one another to form branches and loops. Partial ordering has previously been proposed as a mathematical condition for the consistency of a proposed MSA (Morgenstern *et al.*, 1996). This principle has been used by the program DIALIGN to prune the set of possible combinations of local (segment) alignments while building up an MSA with a greedy algorithm, and has been applied successfully to both DNA and protein sequences (Morgenstern *et al.*, 1996).

By contrast, representing an MSA as a linear structure is equivalent to assuming that all regions of all its sequences are homologous to one another over their entire length (and thus should be aligned, forming one linear sequence of columns). Problems with this assumption are well-known. When standard MSA methods are applied to large sequence families, it is often considered important to 'clip' the sequences before alignment, removing non-homologous parts. In fact, the CLUSTAL manual warns that failure to remove such heterogeneous, non-homologous regions can cause incorrect alignments.

In particular, alignments of multidomain protein families should have complex partial order structure, because such proteins often differ in their domain composition, even when they have one or more domains in common. Figure 5 illustrates that even for alignment of a very small number of proteins (four), the partial order structure can be

complex, interesting, and biologically meaningful. Indeed, generating a POA of the human proteome, and analyzing its branching structure, offers one way of identifying the dictionary of domain types, and cataloguing how they are combined. We anticipate that POA can provide interesting new lines of work for the study of multidomain protein families.

Future work

The simplistic POA algorithm presented in this paper has a number of flaws that can be corrected by combining it with other sequence alignment algorithms. For example, POA is sensitive to sequence alignment order. One way to address this would be to apply a CLUSTAL-like progressive alignment algorithm, using a guide tree to establish the best order for aligning the sequences. Since this requires alignment of one set of aligned sequences to another, this involves extending the POA algorithm to align two PO-MSAs to each other. We will describe this PO-PO algorithm and its application to CLUSTAL-like progressive alignment elsewhere. A second flaw in the simple POA algorithm is that it uses a pairwise alignment scoring matrix as opposed to more sophisticated profile scoring statistics. This becomes a problem when aligning large numbers of highly variable sequences. Since POA allows recombination when searching for an optimal alignment, when there are multiple residue choices at each position in the PO-MSA, POA will pick the path that scores best at each position, and alignment of a random sequence will no longer yield a neutral (zero) score. This can lead to aligning non-homologous regions of sequence, which should not be aligned. This problem can be solved by properly adjusting the alignment scoring for statistical significance, so that alignment of shuffled sequences yields a zero score. Profile-based scoring methods (Thompson *et al.*, 1994) have solved similar statistical problems.

POA is complementary to existing sequence alignment approaches in many areas. For example, the data compression inherent in the PO-MSA representation could be used to develop accelerated forms of search methods like FASTA (Pearson, 1990) and PSI-BLAST (Altschul *et al.*, 1997). By using POA to pre-align homologous sequences in the input sequence database, the search algorithm could use the PO-MSAs as the search database. This could be much faster than searching all the individual sequences separately, especially for datasets with high levels of redundancy (like ESTs), and potentially more sensitive because it can use position-specific profiles (like PSI-BLAST) based on the PO-MSAs, and could detect novel 'recombinations' of different elements from a homologous sequence family. The PO-MSA representation of MSAs is itself also potentially useful, as the basis for applying and developing novel graph

algorithms for finding biologically interesting features in sequence data. Finally, the PO-MSA is fully compatible with probabilistic approaches based on Markov chains, and is a natural object for Hidden Markov Model (HMM) computations (reviewed in Eddy, 1996).

ACKNOWLEDGEMENTS

We wish to thank D.S.Parker, D.Lipman, P.Mallick and D.Eisenberg for their helpful discussions and comments on this work. This work was supported by Department of Energy grant DEFG0387ER60615, National Science Foundation grant IIS-0082964, and a grant from the Searle Scholars Program. C.G. is supported by a Department of Energy Computational Science Graduate fellowship.

REFERENCES

- Altschul,S.F. (1989) Gap costs for multiple sequence alignment. *J. Theor. Biol.*, **138**, 297–309.
- Altschul,S.F., Madden,T.L., Schäffer,A.A., Zhang,J., Zhang,Z., Miller,W. and Lipman,D.J. (1997) Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Res.*, **25**, 3389–3402.
- Barton,G.J. and Sternberg,M.J. (1987) A strategy for the rapid multiple alignment of protein sequences. Confidence levels from tertiary structure comparisons. *J. Mol. Biol.*, **198**, 327–337.
- Boguski,M.S. and Schuler,G. (1995) ESTablishing a human transcript map. *Nature Genet.*, **10**, 369–371.
- Burke,J., Wang,H., Hide,W. and Davison,D.B. (1998) Alternative gene form discovery and candidate gene selection from gene indexing projects. *Genome Res.*, **8**, 276–290.
- Eddy,S.R. (1996) Hidden Markov models. *Curr. Opin. Struct. Biol.*, **6**, 361–365.
- Feng,D.F. and Doolittle,R.F. (1987) Progressive sequence alignment as a prerequisite to correct phylogenetic trees. *J. Mol. Evol.*, **25**, 351–360.
- Gordon,D., Desmarais,C. and Green,P. (2001) Automated finishing with autofinish. *Genome Res.*, **11**, 614–625.
- Gotoh,O. (1993) Optimal alignment between groups of sequences and its application to multiple sequence alignment. *CABIOS*, **9**, 361–370.
- 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.*, **264**, 823–838.
- Higgins,D.G. and Sharp,P.M. (1988) CLUSTAL: a package for performing multiple sequence alignment on a microcomputer. *Gene*, **73**, 237–244.
- Irizarry,K., Kustanovich,V., Li,C., Brown,N., Nelson,S., Wong,W. and Lee,C. (2000) Genome-wide analysis of single-nucleotide polymorphisms in human expressed sequences. *Nature Genet.*, **26**, 233–236.
- Irizarry,K., Hu,G., Wong,M.L., Licinio,J. and Lee,C. (2001) Single nucleotide polymorphism identification in candidate gene systems of obesity. *Pharmacogenomics J.*, **1**, 193–203.
- Lipman,D.J., Altschul,S.F. and Kececioglu,J.D. (1989) A tool for multiple sequence alignment. *Proc. Natl Acad. Sci. USA*, **86**, 4412–4415.

- Mevisen,H.T. and Vingron,M. (1996) Quantifying the local reliability of a sequence alignment. *Protein Eng.*, **9**, 127–132.
- Modrek,B., Resch,A., Grasso,C. and Lee,C. (2001) Genome-wide analysis of alternative splicing using human expressed sequence data. *Nucleic Acids Res.*, **29**, 2850–2859.
- Morgenstern,B., Dress,A. and Werner,T. (1996) Multiple DNA and protein sequence alignment based on segment-to-segment comparison. *Proc. Natl Acad. Sci. USA*, **93**, 12 098–12 103.
- 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.*, **48**, 443–453.
- Notredame,C., Holm,L. and Higgins,D.G. (1998) COFFEE: an objective function for multiple sequence alignments. *Bioinformatics*, **14**, 407–422.
- Notredame,C., Higgins,D.G. and Heringa,J. (2000) T-coffee: a novel method for fast and accurate multiple sequence alignment. *J. Mol. Biol.*, **302**, 205–217.
- Pearson,W.R. (1990) Rapid and sensitive sequence comparison with FASTP and FASTA. *Methods Enzymol.*, **183**, 63–98.
- Smith,T.F. and Waterman,M.S. (1981) Identification of common molecular subsequences. *J. Mol. Biol.*, **147**, 195–197.
- Subbiah,S. and Harrison,S.C. (1989) A method for multiple sequence alignment with gaps. *J. Mol. Biol.*, **209**, 539–548.
- Taylor,W.R. (1986) Identification of protein sequence homology by consensus template alignment. *J. Mol. Biol.*, **188**, 233–258.
- 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.*, **22**, 4673–4680.