Chapter 5
Multiple Sequence Alignment

Aligning simultaneously several sequences is among the most important problems in computational molecular biology. It finds many applications in computational genomics and molecular evolution.

This chapter is divided into five sections. Section 5.1 gives a definition of multiple sequence alignment and depicts a multiple alignment of three sequences.

There are several models for assessing the score of a given multiple sequence alignment. The most popular ones are sum-of-pairs (SP), tree alignment, and consensus alignment. In Section 5.2, we focus our discussion on the SP alignment scoring scheme.

Section 5.3 considers the problem of aligning three sequences based on the SP alignment model. An exact multiple alignment algorithm is given for such a problem.

For many applications of multiple alignment, more efficient heuristic methods are often required. Among them, most methods adopt the approach of "progressive" pairwise alignments introduced in Section 5.4. It iteratively merges the most similar pairs of sequences/alignments following the principle "once a gap, always a gap."

Finally, we conclude the chapter with the bibliographic notes in Section 5.5.

5.1 Aligning Multiple Sequences

Simultaneous alignment of several sequences is among the most important problems in computational molecular biology. Its purpose is to reveal the biological relationship among multiple sequences. For example, it can be used to locate conservative regions, study gene regulation, and to infer evolutionary relationship of genes or proteins.

Recall the definition of a pairwise alignment given in Chapter 1.2.1. An *alignment* of two sequences is obtained by inserting some number (perhaps 0) of spaces, denoted by dashes, in each sequence to yield padded sequences of equal length, then placing the first padded sequence above the other. To emphasize that all sequence

```
S1:   TTATTTCACC-----CTTATATCA
S2:   TCCTTTCA--------TGATATCA
S3:   T--TTTCACCGACATCAGATAAAA
```

Fig. 5.1 A sample of multiple sequence alignment.

entries are required to appear in the alignment, we use the term *global* (as opposed to *local*). Each column of an alignment is called an *aligned pair*. In general, we require that an alignment does not contain two spaces in a column, which we call the *null column*. In context where null columns are permitted the term *quasi-alignment* is used to emphasize that the ban on null columns has been temporarily lifted.

Assume that we are given $S_1, S_2, \ldots, S_m$, each of which is a sequence of "letters." A multiple alignment of these sequences is an $m \times n$ array of letters and dashes, such that no column consisting entirely of dashes, and removing dashes from row i leaves the sequence S_i for $1 \leq i \leq m$. For each pair of sequences, say S_i and S_j, rows i and j of the m-way alignment constitute a pairwise quasi-alignment of S_i and S_j; removing any null columns produces a pairwise alignment of these sequences. Figure 5.1 gives a multiple alignment of three sequences:

5.2 Scoring Multiple Sequence Alignment

For any two given sequences, there are numerous alignments of those sequences. To make explicit the criteria for preferring one alignment over another, we define a score for each alignment. The higher the score is, the better the alignment is. Let us review the scoring scheme given in Section 1.3. First, we assign a score denoted $\sigma(x,y)$ to each aligned pair $\begin{pmatrix} x \\ y \end{pmatrix}$. In the cases that x or y is a space, $\sigma(x,y) = -\beta$. Score function σ depends only on the contents of the two locations, not their positions within the sequences. Thus, $\sigma(x,y)$ does not depend on where the particular symbols occur. However, it should be noted that there are situations where position-dependent scores are quite appropriate. Similar remarks hold for the gap penalties defined below.

The other ingredient for scoring pairwise alignments is a constant *gap-opening penalty*, denoted α, that is assessed for each gap in the alignment; a *gap* is defined as a run of spaces in a row of the alignment that is terminated by either a non-space symbol or an end of the row. Gap penalties are charged so that a single gap of length, say, k will be preferred to several gaps of total length k, which is desirable since a gap can be created in a single evolutionary event. Occasionally, a different scoring criterion will be applied to *end-gaps*, *i.e.*, gaps that are terminated by an end of the row. The score of an alignment is defined as the sum of σ values for all aligned pairs, minus α times the number of gaps.

Selection of the scoring parameters σ and α is often a major factor affecting the usefulness of the computed alignments. Ideally, alignments are determined in such a way that sequence regions serving no important function, and hence evolving freely, should not align, whereas regions subject to purifying selection retain sufficient similarity that they satisfy the criteria for alignment. The chosen alignment scoring scheme determines which regions will be considered non-aligning and what relationships will be assigned between aligning regions. Appropriateness of scoring parameters depends on several factors, including evolutionary distance between the species being compared.

When simultaneously aligning more than two sequences, we want knowledge of appropriate parameters for pairwise alignment to lead immediately to appropriate settings for the multiple-alignment scoring parameters. Thus, one might desire a scoring scheme for multiple alignments that is intimately related to their induced pairwise alignment scores. Of course, it is also necessary that the approach be amenable to a multiple-alignment algorithm that is reasonably efficient with computer resources, *i.e.*, time and space.

There are several models for assessing the score of a given multiple sequence alignment. The most popular ones are sum-of-pairs (SP), tree alignment, and consensus alignment. We focus our discussion on the SP alignment scoring scheme.

To attain this tight coupling of pairwise and multiple alignment scores at a reasonable expense, many multiple alignment tools have adopted the *SP* substitution scores and quasi-natural gap costs, as described by Altschul [2]. Some notation will help for a precise description of these ideas.

Scores for multiple alignments are based on pairwise alignment scores, which we described above. With an m-way alignment Π, we would like to determine appropriate parameters for the score, say $Score_{i,j}$, for pairwise alignments between S_i and S_j (*i.e.*, the ith and jth sequences), then set

$$(SP)\ Score(\Pi) = \sum_{i<j} Score_{i,j}(\Pi_{i,j}),$$

where $\Pi_{i,j}$ is the pairwise alignment of S_i and S_j induced by Π (see Figure 5.2).

```
Π1,2
    S1:   TTATTTCACCCTTATATCA
    S2:   TCCTTTCA---TGATATCA

Π1,3
    S1:   TTATTTCACC-----CTTATATCA
    S3:   T--TTTCACCGACATCAGATAAAA

Π2,3
    S2:   TCCTTTCA--------TGATATCA
    S3:   T--TTTCACCGACATCAGATAAAA
```

Fig. 5.2 Three pairwise alignments induced by the multiple alignment in Figure 5.1.

```
S1:  TTATTTCACC-----CTTATATCA
S2:  TCCTTTCA--------TGATATCA
```

Fig. 5.3 A quasi-alignment of Π (in Figure 5.1) projected on S_1 and S_2 without discarding null columns.

The projected substitution costs of SP-alignments can be computed easily. However, as noted in Altschul [2], strictly computing the imposed affine gap costs results in undesirable algorithmic complexity. The complications come from the fact that we may have to save a huge number of the relevant histories in order to decide if we need to charge a gap opening-up penalty for a given deletion (or insertion) pair. Altschul further observed that this complexity of saving all possible relevant histories can be reduced dramatically if for every pair of rows of the m-way alignment we assess an additional gap penalty for each "quasi-gap," defined as follows. Fix a pair of rows and consider a gap, G, in the corresponding pairwise quasi-alignment, i.e., a run of consecutive gap symbols occurring in one of the rows (the run should be extended in both directions until it hits a letter or the end of the sequence). If at least one space in G is aligned with a letter in the other row, then G corresponds to a gap in the pairwise alignment (i.e., after discarding null columns), and hence is penalized. The other possibility is that every space in G is aligned with a space in the other sequence. If the gap in the other sequence starts strictly before and ends strictly after G, then G is called a *quasi-gap* and is penalized. For example, the gap in S_2 of Figure 5.3 is a quasi-gap in a projected alignment without discarding null columns. In $\Pi_{1,2}$ of Figure 5.2, there is only one deletion gap counted. But in practical implementation, we might assess two gap penalties since an additional quasi-gap penalty might be imposed. If either end of G is aligned to an end of the gap in the other sequence, then the gap is not penalized.

In summary, a multiple alignment is scored as follows. For each pair of rows, say rows i and j, fix appropriate substitution scores $\sigma_{i,j}$ and a gap cost $\alpha_{i,j}$. Then the score for the multiple alignment is determined by equation (SP), where each $Score_{i,j}(\Pi_{i,j})$ is found by adding the σ values for non-null columns of the pairwise quasi-alignment, and subtracting a gap penalty α for each gap and each quasi-gap.

5.3 An Exact Method for Aligning Three Sequences

The pairwise alignment algorithms introduced in Chapter 3 can be easily extended to align for more than two sequences. Consider the problem of aligning three sequences $A = a_1a_2\ldots a_{n_1}$, $B = b_1b_2\ldots b_{n_2}$, and $C = c_1c_2\ldots c_{n_3}$ based on the SP alignment model. Let x, y and z be any alphabet symbol or a gap symbol. Assume that a simple scoring scheme for pairwise alignment is imposed where a score $\chi(x,y)$ is defined for each aligned pair $\begin{pmatrix} x \\ y \end{pmatrix}$. Let $\phi(x,y,z)$ be the score of an aligned col-

umn $\begin{pmatrix} x \\ y \\ z \end{pmatrix}$. The score $\phi(x,y,z)$ can be computed as the sum of $\chi(x,y)$, $\chi(x,z)$, and $\chi(y,z)$.

Let $S[i,j,k]$ denote the score of an optimal alignment of $a_1a_2\ldots a_i$, $b_1b_2\ldots b_j$, and $c_1c_2\ldots c_k$. With proper initializations, $S[i,j,k]$ for $1 \le i \le n_1$, $1 \le j \le n_2$, and $1 \le k \le n_3$ can be computed by the following recurrence.

$$S[i,j,k] = \max \begin{cases} S[i-1,j,k] + \phi(a_i,-,-), \\ S[i,j-1,k] + \phi(-,b_j,-), \\ S[i,j,k-1] + \phi(-,-,c_k), \\ S[i,j-1,k-1] + \phi(-,b_j,c_k), \\ S[i-1,j,k-1] + \phi(a_i,-,c_k), \\ S[i-1,j-1,k] + \phi(a_i,b_j,-), \\ S[i-1,j-1,k-1] + \phi(a_i,b_j,c_k). \end{cases}$$

The value $S[n_1,n_2,n_3]$ is the score of an optimal multiple alignment of A, B, and C. The three-dimensional dynamic-programming matrix contains $O(n_1n_2n_3)$ entries, and each entry takes the maximum value from the $2^3 - 1 = 7$ possible entering edges. All possible combinations of ϕ values can be computed in advance. Thus, we can align three sequences of lengths n_1, n_2 and n_3 in $O(n_1n_2n_3)$ time.

Following this approach, one can easily derive an $O(n^m2^m)$-time algorithm for constructing m-way alignment of length n. This exact method in general requires too much time and space to be practical for DNA sequences of average length. Not to mention that there are a lot more possible entering edges (configurations) for each entry if affine gap penalties or affine quasi-gap penalties are used. Furthermore, the multiple sequence alignment has been shown to be NP-hard by Wang and Jiang [195], meaning that there is no polynomial-time algorithm for it unless NP=P.

Despite the intractability of the multiple alignment problem, some researchers proposed "efficient" exact methods by pruning the dynamic-programming matrix with some optimal score lower bound. These exact methods have been proved to be useful in certain context.

5.4 Progressive Alignment

For many applications of multiple alignment, more efficient heuristic methods are often required. Among them, most methods adopt the approach of "progressive" pairwise alignments proposed by Feng and Doolittle [69]. It iteratively merges the most similar pairs of sequences/alignments following the principle "once a gap, always a gap." Thus, later steps of the process align two "sequences," one or both of which can themselves be an alignment, *i.e.*, sequence of fixed-height columns.

In aligning two pairwise alignments, the columns of each given pairwise alignment are treated as "symbols," and these sequences of symbols are aligned by

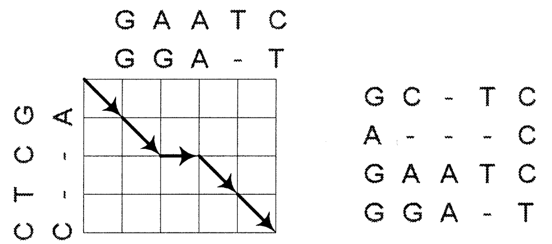

Fig. 5.4 Aligning two alignments.

padding each sequence with appropriate-sized columns containing only dash symbols. It is quite helpful to recast the problem of aligning two alignments as an equivalent problem of finding a maximum-scoring path in an alignment graph. For example, the path depicted in Figure 5.4 corresponds to a 4-way alignment. This alternative formulation allows the problem to be visualized in a way that permits the use of geometric intuition. We find this visual imagery critical for keeping track of the low-level details that arise in development and implementation of alignment algorithms.

Each step of the progressive alignment procedure produces an alignment that is highest-scoring relative to the chosen scoring scheme subject to the constraint that columns of the two smaller alignments being combined are treated as indivisible "symbols." Thus, the relationships between entries of two of the original sequences are fixed at the first step that aligns those sequences or alignments containing those sequences.

For that reason, it is wise to first compute the pairwise alignments that warrant the most confidence, then combine those into multiple alignments. Though each step is performed optimally, there is no guarantee that the resulting multiple alignment is highest-scoring over all possible ways of aligning the given sequences. An appropriate order for progressive alignment is very critical for the success of a multiple alignment program. This order can either be determined by the guide tree constructed from the distance matrix of all pairs of sequences, or can be inferred directly from an evolutionary tree for those sequences. In any case, the progressive alignment algorithm invokes the "generalized" pairwise alignment $m-1$ times for constructing an m-way alignment, and its time complexity is roughly the order of the time for computing all $O(m^2)$ pairwise alignments.

5.5 Bibliographic Notes and Further Reading

In spite of the plethora of existing ideas and methods for multiple sequence alignment, it remains as an important and exciting line of investigation in computational

molecular biology. Recently, Miller et al. [140] compiled a set of alignments of 28 vertebrate genome sequences in the UCSC Genome Browser [104].

5.1

We compile a list of multiple alignment tools in Table C.3 of Appendix C. There are a few multiple alignment benchmarks available for testing such as BAliBASE [20], PREFAB [62], and SMART [122]. Thompson et al. [190] gave the first systematic comparison of multiple alignment programs using BAliBASE benchmark dataset.

5.2

A recent survey by Notredame [154] divides the scoring schemes in two categories. Matrix-based methods, such as ClustalW [120, 189], Kalign [121], and MUSCLE [62], use a substitution matrix to score matches. On the other hand, consistency-based methods, such as T-Coffee [155], MUMMALS [163], and ProbCons [59], compile a collection of pairwise alignments and produce a position-specific substitution matrix to judge the consistency of a given aligned pair. An alternative is to score a multiple alignment by using a consensus sequence that is derived from the consensus of each column of the alignment.

Besides these two categories, there are some other scoring schemes. For example, DIALIGN [143] focuses on scoring complete segments of sequences.

5.3

A straightforward multiple sequence alignment algorithm runs in exponential time. More "efficient" exact methods can be found in [38, 84, 127]. In fact, it has been shown to be NP-hard by Wang and Jiang [195]. Bafna et al. [18] gave an algorithm with approximation ratio $2-\ell/m$ for any fixed ℓ.

5.4

A heuristic progressive alignment approach was proposed by Feng and Doolittle [69]. It iteratively merges the most similar pairs of sequences/alignments following the principle "once a gap, always a gap."

Surprisingly, the problem of aligning two SP-alignments under affine gap penalties has been proved to be NP-hard [107, 132]. However, it becomes tractable if affine quasi-gap penalties are used [2].

ClustalW [120, 189] is the most popular multiple alignment program. It works in three stages. In the first stage, all pairs of sequences are aligned, and a distance matrix is built. The second stage constructs a guide tree from the distance matrix. Finally, in the third stage, the sequences as well as the alignments are aligned progressively according to the order in the guide tree.

YAMA [41] is a multiple alignment program for aligning long DNA sequences. At each progressive step, it implements the generalized Hirschberg linear-space algorithm and maximizes the sum of pairwise scores with affine quasi-gap penalties. To increase efficiency, a step of the progressive alignment algorithm can be constrained to the portion of the dynamic-programming grid lying between two boundary lines. Another option is to consider constrained alignments consisting of aligned pairs in nearly optimal alignments. A set of "patterns" is specified (perhaps the consensus sequences for transcription factor binding sites); YAMA selects, from among all alignments with the highest score, an alignment with the largest number of conserved blocks that match a pattern.

MUSCLE [62, 63] is a very efficient and accurate multiple alignment tool. It uses a strategy similar to PRRP [80] and MAFFT [106]. It works in three stages. Stage 1 is similar to ClustalW. A guide tree is built based on the distance matrix, which is derived from pairwise similarities. A draft progressive alignment is built according the constructed guide tree. In Stage 2, MUSCLE computes a Kimura distance matrix [111] using the pairwise identity percentage induced from the multiple alignment. It then builds a new guide tree based on such a matrix, compares it with the previous tree, and re-aligns the affected subtrees. This stage may be iterated if desired. Stage 3 iteratively refines the multiple alignment by re-aligning the profiles of two disjoint subsets of sequences derived from deleting an edge from the tree constructed in Stage 2. This refinement is a variant of the method by Hirosawa et al. [90]. If the new alignment has a better score, save it. The refinement process terminates if all edges incur no changes or a user-specified maximum number of iterations has been reached.

PART II. THEORY