3 Comparative Sequence Analysis

Comparative sequence analysis has been a powerful tool in bioinformatics for addressing a variety of issues. Applications range from grouping of sequences (i.e. protein sequences into families) to *de novo* pattern discovery of functional signatures. Sequence comparison aims at detecting similarities between sequences. Consequently, one has to think about ways to compare sequences, rank the results by some score function and statistically assess the significance of an alignment result. Most of the fundamentals presented here are adapted from excellent introductory text books like Durbin et al. (1998) and Pevzner (2000).

3.1 Sequence Alignment

The most basic analysis task is to ask if two sequences are related. This is usually done by aligning the two sequences (**pairwise alignment**). Alignments will be discussed in the sense of aligning two sequences unless stated differently. Mutation in DNA is a natural evolutionary process: DNA replication errors cause substitutions, insertions and deletions of nucleotides, leading to change in textual information ("editing").

Most DNA sequence comparison algorithms use a set of three editing operations. The three common operations are exemplified on a sequence $V = abc$:

- **Substitution.** A symbol at a given sequence position is transformed into another one. $abc \rightarrow adc$.
- **Insertion.** A symbol is inserted into a sequence. $abc \rightarrow abdc$.
- **Deletion.** A symbol is deleted from a sequence. $abc \rightarrow ac$.

Insertions and deletions are represented by **gap characters** `'-'` in an alignment. The presented operations are local changes that affect only one or few bases.

Example 3.1. The alignment $(d \rightarrow \varepsilon, a \rightarrow a, \varepsilon \rightarrow i, r \rightarrow r, l \rightarrow l, i \rightarrow i, n \rightarrow n, g \rightarrow e)$ is displayed as follows:

u: `da-rling`
v: `-airline`

where ε is the empty character and transitions from or to the empty character are denoted as insertions and deletions, respectively.

Larger changes in genetic sequence (i.e. chromosomal rearrangements) place DNA chunks of several thousands to millions of nucleotides into a new genomic context. These large-scale rearrangements are often observed in cancer cells where entire chromosomes are broken up and randomly fused. Figure 3.1 summarizes the most important of these changes operating at the chromosome level. Intriguingly, new evidence suggests that the same rearrangements happen on a smaller scale even in promoter regions (Wray et al., 2003).

3.1.1 Global vs. Local sequence alignment

The similarity between entire sequences was the first alignment problem that caught the attention of researchers. Needleman and Wunsch (1970) were the first to report a **global alignment approach** to biological sequences. A global alignment is meaningful for closely related sequence, i.e., members of the same protein family, such as *globins*, that are highly conserved and have almost no variation in sequence length. However, the score of an alignment between substrings of two sequences may be larger than the overall score in a global alignment. In this case, detecting local sequence similarities is more informative. A good example to illustrate this is the *homeodomain* (Figure 2.5). The homeodomain is the DNA binding domain of many transcription factors, which are involved in developmental processes. Although the overall sequence similarity is very low among homeodomain containing proteins from different species, the homeodomain itself is highly conserved among all candidates.

Smith and Waterman (1981) proposed an algorithmic solution for the **local alignment problem**. Both algorithms return optimal solutions to the respective problems. Both employ **dynamic programming** for this purpose. Dynamic programming can be briefly defined as an algorithmic technique in which an optimization problem is solved by caching subproblem solutions (memorization) rather than recomputing them.

3.1.2 Models of nucleotide substitution

In pairwise sequence comparisons, sixteen different types of nucleotide pairs may occur at aligned positions: four **identical nucleotide pairs**, four **transition-type pairs** (P) and eight **transversion-type pairs** (Q). Transitions are interchanges between pyrimidines, or between purines. Transversions are interchanges between purines and pyrimidines. The number of nucleotide substitutions $d = P + Q$ can be estimated from the observed number of substitutions $\hat{d}$. However, $\hat{d}$ seriously underestimates the true d, because it does not take into account backward and parallel mutations. To get

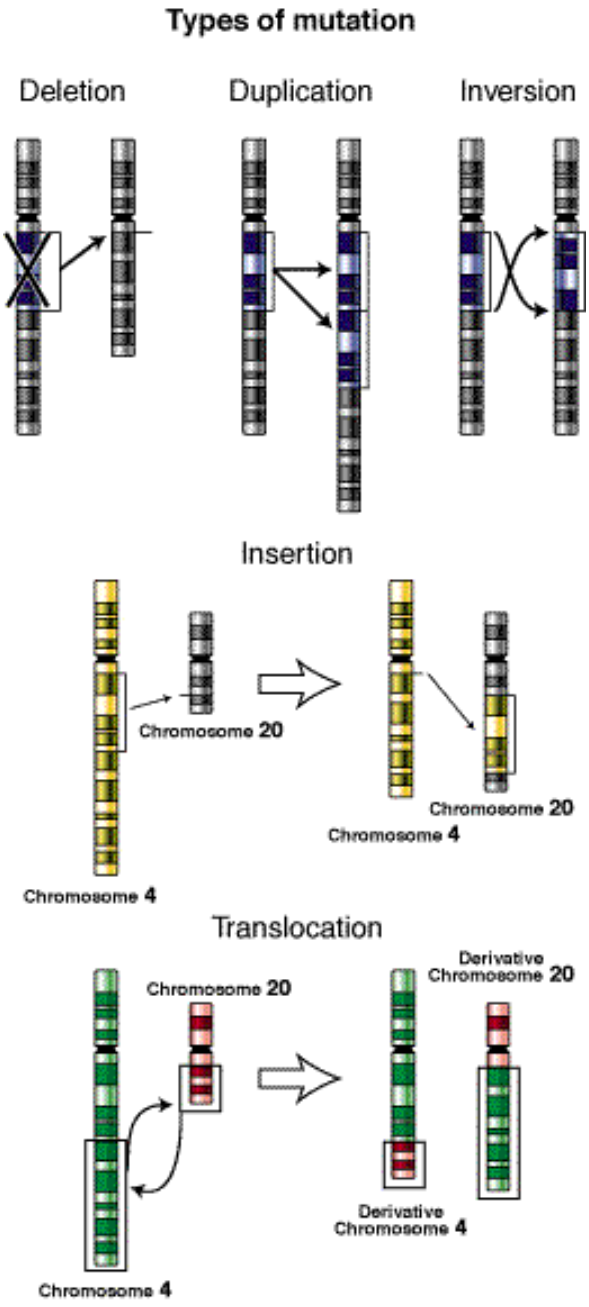

Figure 3.1: Chromosomal rearrangements. Large-scale edit operations may alter the structure of a genome (i.e. number of chromosomes) and place large chunks of DNA into a new genomic context. Image reproduced with permission from http://www.accessexcellence.org/AB/GG/mutation2.html, The National Health Museum, USA.

a more accurate idea of the true amount of sequence change during the evolutionary history of sequences, many mathematical models have been proposed.

	Jukes-Cantor model				Kimura model			
	A	C	G	T	A	C	G	T
A	$-$	α	α	α	$-$	β	α	β
C	α	$-$	α	α	β	$-$	β	α
G	α	α	$-$	α	α	β	$-$	β
T	α	α	α	$-$	β	α	β	$-$

Table 3.1: Models of nucleotide substitution.

We will now review the two most simple models of nucleotide sequence evolution: the one of Jukes and Cantor (1969) and the one of Kimura (1980), which is utilized in our promoter sequence comparisons.

Jukes-Cantor Model The basic assumption is equality of substitution frequency for any nucleotide at any site. Thus, changing a nucleotide to each of the three remaining nucleotides has probability α per time unit. The rate of nucleotide substitution per site per time unit is then $r = 3\alpha$. Let q be the proportion of identical nucleotides between two sequences. In a continuous time model q is given by the following equation:

$$q = 1 - \frac{3}{4}\left(1 - \exp^{-8rt/3}\right) \tag{3.1}$$

The expected number of substitutions per site (d) is approximately $2rt$. Rearranging the equation above yields:

$$d = -\frac{3}{4}\ln\left[1 - \frac{4}{3}p\right] \tag{3.2}$$

where $p = 1 - q$ is the proportion of different nucleotides.

Kimura model In DNA, the rate of transitions is usually higher than that of transversions. In the model of Kimura, both types of substitution rates are explicitly modeled with parameters α (the rate of transitions) and β (the rate of transversions). The total substitution rate per site and time unit is then $\alpha + 2\beta$. Hence, the expected number of nucleotide substitutions is given by $d \equiv 2rt = 2\alpha t + 4\beta t$ where t is the time after divergence of two sequences. Using his model, Kimura showed that the frequencies of Transitions (T_s) and Transversions (T_v) are given by

$$T_s = \frac{1}{4}\left(1 - 2\exp^{-4(\alpha+\beta)t} + \exp^{-8\beta t}\right) \tag{3.3}$$

$$T_v = \frac{1}{2}\left(1 - \exp^{-8\beta t}\right) \tag{3.4}$$

The expected number of substitutions per site (d) is then

$$d = -\frac{1}{2}\ln\left[1 - 2T_s - T_v\right]\ln\left[1 - 2T_v\right] \tag{3.5}$$

Both models have an equilibrium frequency of each nucleotide of 0.25. Both models are reversible meaning that sequences evolve equally over time. In this regard, it is irrelevant whether some sequence A evolves into B or vice versa.

For this thesis work, we refrain from using time units as measured in years. We are more interested in expressing the expected degree of similarity between two sequences expressed in the expected number of substitutions per site (d).

Therefore, we employ the notion of point accepted mutation per site or PAM.

Definition 3.2 (PAM)**.** One point accepted mutation (1 PAM) is defined as an expected number of substitutions per site of 0.01. A 1 PAM substitution matrix is thus derived from any evolutionary model by setting the row sum of off-diagonal terms to 0.01 and adjusting the diagonal terms to keep the row sum equal to 1.

A substitution matrix M for any PAM distance n is then obtained by iterative multiplication of a 1 PAM matrix: $M_n = (M_1)^n$. We are now able to model substitution processes by selecting an evolutionary model and a PAM distance, which reflects the expected degree of sequence similarity.

3.1.3 How to score an alignment.

We are still missing a vital part to compute an alignment. As previously mentioned, an alignment problem is an optimization problem and thus requires an objective function that could be maximized in terms of an **alignment score**.

The total score we assign to an alignment will be the sum of scores for each aligned pair of residues and each gap. If the two sequences under comparison are related, we expect a match between identical residue pairs to be more likely than we expect by their single frequencies. Thus matches should contribute positively to the alignment score whereas non-conservative substitutions (i.e. transversions) and gaps should penalized. Generally, an additive scoring scheme is used under the assumption that mutations occur independently. Most alignment algorithms depend on this assumption.

Definition 3.3 (Substitution score)**.** Assume **s** and **t** are two sequence of length n and m over an alphabet $\Sigma := \{A, C, G, T\}$. Given a gapless alignment of those sequences, we want to assign a score to the alignment that gives a measure of the relative likelihood that the sequences are related as opposed to being unrelated. A prominent additive scoring system is based on the **log-odds ratio** of a residue pair (s_i,t_j) occurring as an aligned pair instead of being unaligned.

$$Score = \sum_i pairscore(s_i, t_j) \tag{3.6}$$

where

$$\texttt{pairscore}(a, b) = \log(\frac{p_{ab}}{q_a q_b}) \tag{3.7}$$

p_{ab} is the probability of seeing a residue pair (a, b) in a *match* model (i.e. a 1 PAM Kimura model) and q_a, q_b are the letter frequencies in a random model assuming that letters occur independently.

Introducing gaps in alignments raises the question of evaluating them. Since most alignment algorithms employ **affine gap penalties**, we will focus on those.

Definition 3.4 (Affine gap penalties)**.** For affine gap penalties, the score for a gap of length x is $-(g + ex)$, where $g > 0$ is the penalty for introducing a gap (**gap open**) and $e > 0$ is the penalty for each gap symbol (**gap extension**).

Despite the fact that affine gap penalties are the most commonly used, various applications exist where non-linear gap penalties may be advantageous over the presented ones (Mott, 1999).

3.1.4 Alignment algorithms

Now that we have a scoring system, this section outlines the practical steps to obtain alignments of different types. All presented algorithms make use of dynamic programming. Dynamic programming algorithms are guaranteed to find the optimal scoring alignment or set of alignments. On the other hand, fast *heuristic* methods have been developed for aligning sequences. However, these are not guaranteed to find the optimal alignment, but produce high-quality results in the vast majority of trials.

Global alignment with gaps Two sequences are aligned over their entire lengths, allowing gaps. Thus, the alignment score is the sum of all pair scores. If the lengths of sequences differ, additional gaps are inserted. Gotoh (1982) proposed an improved version of the global alignment algorithm. An optimal alignment is built using previous optimal solutions of smaller subsequences.

Local alignment with gaps A global alignment strategy is not suitable in many situations. In the age of databases, one would like to scan a new piece of genomic sequence versus a repository of reference sequences. A better way of doing that would be to look for similarities among subsequences of the query and database sequences. Thus, the only difference to global alignment is that the optimal alignment can start and stop anywhere in the two sequences a, b of length n, m as long as the alignment coordinates $(a_i, b_j)(a_k, b_l)$ obey $i < k \leq n$ and $j < l \leq m$. Smith and Waterman (1981) introduced an algorithm for this particular task.

Suboptimal local alignments The best local alignment of two DNA sequences does not always capture the biological meaning. Recall the situation of Figure 2.7 where the primary goal was to identify regions of local similarity. These regions could be regulatory elements (as shown) or exons in a gene finding task. The best local alignment would be either misleading (e.g. returns only one particular element) or non-specific (e.g. covers regions of poor conservation) in such a setting. A better way of handling such comparisons is to retrieve more than one alignment from the alignment space. Waterman and Eggert (1987) proposed an algorithm for finding non-trivial local similarities, which are called suboptimal local alignments. This algorithm is of fundamental importance to this thesis and will be discussed in a separate section. It does not constrain the position of alignments other than prohibiting multiple occurrences of the same residue pairings in different alignments.

Heuristic approaches The previously presented algorithms are guaranteed to find the optimal solution for a given scoring scheme. However, these algorithms are not a feasible solution to the comparison of long sequences in the order of 100 Kb to several Mb. Speed (not memory) becomes an issue here. Heuristic approaches mitigate this problem by trying to reduce the search space, while still maintaining a high sensitivity level.

The most famous tool is certainly the BLAST package (Altschul et al., 1990). The BLAST algorithm exploits the idea that meaningful alignments contain short identical subsequences, or very high scoring matches. BLAST compiles a list of all words of a fixed length (e.g. 11 nucleotides for DNA), that would match the query sequence with scores higher than some threshold. BLAST assumes collinearity of the local similarities.

Recent developments Heuristic algorithms for large-scale comparisons of genomic regions have emerged as a new field in computational biology (Miller, 2001). Ureta-Vidal et al. (2003) give a recent survey of the field and list all available "genome alignment" tools. These software packages are readily applicable to compare regions of genomes showing conserved synteny. BLASTZ (Schwartz et al., 2003) is the most

similar heuristic solution to the Waterman-Eggert approach since it computes sub-optimal local alignments with gaps. These alignments have no constraints on their position.

3.1.5 The Waterman-Eggert algorithm

The Waterman-Eggert algorithm repeatedly samples optimal local alignments from an alignment matrix H that is updated after each sampling step. The procedure can be summarized as follows:

1. **Initialization.** $C = \{\}$, where C is the set of alignment clumps.
2. **Calculate alignment.** Calculate and output the optimal local alignment i in the alignment matrix H. Same as in Smith and Waterman (1981).
3. **Update alignment matrix** Calculate the alignment clump C_i of an alignment i.
4. **Add clump to set.** $C = C \cup C_i$.
5. **Goto step 2.** Loop until the required number of alignments has been output or the matrix has been exhausted.

Definition 3.5 (Alignment clump)**.** An alignment clump is the part of an alignment matrix that is influenced by the current optimal alignment. An alignment clump contains all residue pairs of the optimal alignments and all "neighboring" matrix cells that were updated (Step 3 above).

Updating procedure The alignment matrix H must be recalculated after sampling from it since one is not interested in reporting the same alignment or trivial alternatives twice. The matrix cell at the start point of the previous alignment is set to zero. All reported residue pairs of the previous alignment are updated according to a new Recurrence Relation (below) where d is the gap cost function.

$$H^*_{i,j} = \max \left\{ \begin{array}{l} H^*_{i-1,j} - d \\ H^*_{i,j-1} - d \\ 0 \end{array} \right\} \tag{3.8}$$

Non-overlapping alignments should not share an aligned residue pair. As a consequence, Recurrence relation 3.8 does not allow a repetition of the alignment of these residue pairs. As soon as a position is reached whose updated entry equals the former value of the matrix cell, the calculation is stopped for the corresponding row. The updating ends after processing the final row of the alignment clump.

The SIM implementation Huang and Miller (1991) were mainly concerned about memory requirements in finding local sequence similarities. They implemented a linear-space version of the Waterman-Eggert algorithm that preserved its time-efficiency. A modification of this implementation was employed for most sequence comparisons of this thesis work. We obtained the source code of the original implementation from http://globin.cse.psu.edu/html/software.html. We modified the program with respect to significance assessment of alignments, file handling and scoring options.

3.2 Alignment statistics

Distinguishing between "true" alignments (those resulting from homology) and spurious alignments (resulting from chance similarities) is a notable issue that we will address in this chapter. Previous biological knowledge on beneficial residue pairs can be accommodated in the scoring scheme of an alignment. Given a suitable scoring scheme, the next question is: what alignment scores are likely to have arisen by chance alone (i.e. by comparing two random sequences). This issue is indeed very pressing in database searches where a large number of sequences are compared to a single query sequence. In short, we define the significance of an alignment as follows:

Definition 3.6 (Significance of an alignment). The significance of an alignment (p-value) is the probability that an equal or better alignment score could be attained by aligning two unrelated (random) sequences. Generally, the p-value of a random variable T is the probability $\mathbb{P}(T \leq t_{observed})$.

With the above definition, we could naively start to generate scores from random sequences and compute p-values for our observed alignments based on those. This simulation approach has the disadvantages that the sample size directly affects the accuracy of the computed p-values. Hence, we need further insights from mathematical theory to tackle this problem. Pagni and Jongeneel (2001) give an introduction on score statistics, which is highly accessible to readers with a non-mathematical background.

3.2.1 Ungapped alignment statistics

The score of a match to a random sequence is the sum of many similar random variables (see Section 3.1.3, additive scoring scheme). Thus, random scores can be well approximated by a normal distribution. Since we are interested in the optimal alignment (highest score), we need to know how the maximum M_N of a series of N

independent normal random variables is distributed. The asymptotic distribution is known as **extreme value distribution (EVD)** and has the form

$$\mathbb{P}(M_N > x) \simeq 1 - \exp^{-KN \exp^{\lambda(x-\mu)}} \tag{3.9}$$

for some constants K and λ.

The right tail of the EVD decreases almost linearly when plotted in a semi-logarithmic coordinate system. Hence, it can be approximated with a decreasing exponential

$$\mathbb{P}(M_N > x) \simeq \exp^{-\lambda(x-\mu)}; \mu \ll x \tag{3.10}$$

In practical applications like database searches, approximation (3.10) is often used for the computation of E-values (the expected number of random hits in a database of a given size at some significance level).

Random models for DNA sequences So far, we refrained from defining a random model for DNA sequences. Traditionally, one would generate a random DNA sequence by sampling letters from an alphabet $\mathcal{A}$ with probabilities $\sum_{i=A,C,G,T} p_i = 1$. However, this is a rather simplistic view of randomness in a biological setting. Global sequence properties like length and nucleotide composition are preserved, but regions of low complexity and local compositional biases are not represented. Previous work in our department (Cusack, 2001) demonstrated that reversing one nucleotide sequence in a pairwise comparison is the most practical solution. Of course, known repeat regions (i.e. *palindromic sequences*) are excluded from a pairwise comparison since they constitute a biological signal.

Local alignment without gaps Karlin and Altschul (1990) showed that the distribution of alignment scores for two sufficiently large sequences tends towards an EVD. They also presented an analytical solution to determine the parameters of the EVD (λ, μ and K) based on known variables (size of search space, residue frequencies and scoring scheme). However, the scoring scheme has to satisfy the condition that the expected score of a random residue pair is below zero.

Number of scores exceeding a threshold What does the distribution of the number of random alignment scores higher than some threshold look like? Assuming an EVD distribution of the maximum scores of random alignments, exceeding a high enough threshold is a rare event. Karlin and Altschul (1990) provided evidence that the number of alignment scores exceeding some threshold t can be modeled by a Poisson distribution under the condition of a negative expected score. In other words, the

expected number of alignments exceeding some ascending score threshold decreases exponentially.

3.2.2 Gapped alignment statistics

So far, no similar analytical solution has been found for alignments containing gaps. However, the same intuition is carried over to gapped alignments. Waterman and Vingron (1994) demonstrated that the number of local gapped alignments exceeding a threshold t (N_t) has a Poisson distribution with mean γmnp^t where mn is the length of the sequence search space and parameters γ and p are unknown for the gapped case. Both parameters can be estimated from *linear regression analysis* either by direct estimation or 'declumping estimation'.

$$\log\ N_t = \log(\gamma mn) + t\ \log\ p \tag{3.11}$$

where $\log\ p$ is the slope and $\log(\gamma mn)$ is the intersection of the regression line.

Phase transition in growth of gapped alignment scores Parameter estimation by regression analysis is only feasible if alignment scores grow logarithmically. This is mainly dependent on the choice of gap costs. If the gap costs are set too low, scores will grow linearly as gap characters become inserted frequently. Long alignments will now be favored over shorter ones simply because they accommodate more matching characters. Alignments, which were computed in the linear phase, lack any biological meaning. The transition between the linear and logarithmic phase is abrupt and is a function of the gap penalty.

Significance of suboptimal alignments Given two sequences, we are now able to assess the significance of the optimal local alignment. However, we still have no way of dealing with suboptimal local alignments. A *priori* the number of conserved segments in a pair of sequences is not known. Therefore, we need a statistical handle on how many local alignments should be accepted.

This problem can be pragmatically approached by choosing a fixed p-value cutoff for all observed alignments. The statistical significance of suboptimal alignments can then be computed in two ways:

Order statistics The idea behind order statistics is to evaluate the r-best alignment score based on the distribution of the r-best random scores. Waterman and

Vingron (1994) showed that this is most conveniently done by Poisson approximation.

$$\lambda = \gamma mnp^t \tag{3.12}$$

$$\mathbb{P}(Score_r \leq t) = \exp(\lambda) \sum_{j=0}^{r-1} \frac{(\lambda)^j}{j!} \tag{3.13}$$

$$\mathbb{P}(Score_r > t) = 1 - \mathbb{P}(Score_r \leq t) \tag{3.14}$$

Suboptimal alignments are accepted if the p-value derived from Equation 3.14 is below some preset threshold. Cusack (2001) recommends a p-value cutoff of 0.001.

Sum statistics An alternative way is to look at the distribution of the sum of scores of suboptimal alignments $H_1, .., H_r$. Each score H_i is normalized such that $H'_i = \frac{H_i}{\log(p)} - \log(\gamma mn)$. Let

$$T_r = \sum_{i=1}^{r} H'_i \tag{3.15}$$

Karlin and Altschul (1993) showed that

$$\mathbb{P}(T_r > t) \approx \frac{\exp(-t)t^{r-1}}{r!(r-1)!} \tag{3.16}$$

An advantage of the sum statistics over the order statistics is that the former weighs the alignments according to their individual scores.

3.3 Multiple alignments

Dynamic programming based alignment algorithms can be extended towards aligning more than two sequences. This is a very costly step since computation time increases exponentially with the number of sequences. Several approaches have been proposed to cut down on memory requirements and computation time. In the program `MSA` by Lipman et al. (1989), which was further improved by Gupta et al. (1995), bounds are defined within which an optimal alignment can be found. This restricted hyper-volume of the multidimensional space is then employed to compute the optimal solution efficiently. Lee et al. (2002) expanded on this issue by applying partial order graphs (POG) to the multiple alignment problem.

Definition 3.7 (Partial order graph). Partial order graphs belong to the class of directed acyclic graphs (DAGs). A DAG is a graph consisting of a set of nodes N and edges E, which are one-way edges and form no cycles. This means that if there is a route from node a to node b then there is no way back. The term partial order refers

to the following properties: Reflexivity ($a \preceq a$ for all $a \in N$), Antisymmetry ($a \preceq b$ and $b \preceq a$ implies $a = b$) and Transitivity ($a \preceq b$ and $b \preceq c$ then $a \preceq c$).

Lee et al. (2002) reduced the problem to subsequent alignment steps of individual sequences to a growing multiple alignment graph (Partial Order Alignment). If the sequences to be aligned share substantial sequence similarity, the number of bifurcation points within the POG stays low and allows rapid computation of the multiple alignment. However, alignment results are sensitive to the input order of sequences.

We will not discuss any heuristic solutions to the multiple alignment problem since we did not employ them in the context of this thesis. A good general summary on multiple alignment algorithms and scoring methods can be found in Nicholas et al. (2002).

Multiple alignment scores There is a long standing debate on the issue of scoring multiple alignments and we refrain from giving an overview here. Luckily, scoring alignments of single sequences to growing POGs is analogous to scoring pairwise alignments. 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 POG, and M is the length of a new linear sequence being aligned.