6

Phylogenetic inference using maximum likelihood methods

THEORY

Heiko A. Schmidt and Arndt von Haeseler

6.1 Introduction

The concept of ***likelihood*** refers to situations that typically arise in natural sciences in which given some data **D**, a decision must be made about an adequate explanation of the data. Thus, a specific model and a hypothesis are formulated in which the model as such is generally not in question. In the phylogenetic framework, one part of the model is that sequences actually evolve according to a tree. The possible hypotheses include the different tree structures, the branch lengths, the parameters of the ***model of sequence evolution***, and so on. By assigning values to these elements, it is possible to compute the probability of the data under these parameters and to make statements about their plausibility. If the hypothesis varies, the result is that some hypotheses produce the data with higher probability than others. Coin-tossing is a standard example. After flipping a coin $n = 100$ times, $h = 21$ heads and $t = 79$ tails were observed. Thus, $\mathbf{D} = (21, 79)$ constitutes a sufficient summary of the data. The model then states that, with some probability, $\theta \in [0, 1]$ heads appear when the coin is flipped. Moreover, it is assumed that the outcome of each coin toss is independent of the others, that θ does not change during the experiment, and that the experiment has only two outcomes (head or tail). The model is now fully specified. Because both, heads and tails, were obtained, θ must be larger than zero and smaller than 1. Moreover, any probability textbook explains

The Phylogenetic Handbook: a Practical Approach to Phylogenetic Analysis and Hypothesis Testing, Philippe Lemey, Marco Salemi, and Anne-Mieke Vandamme (eds.). Published by Cambridge University Press. © Cambridge University Press 2009.

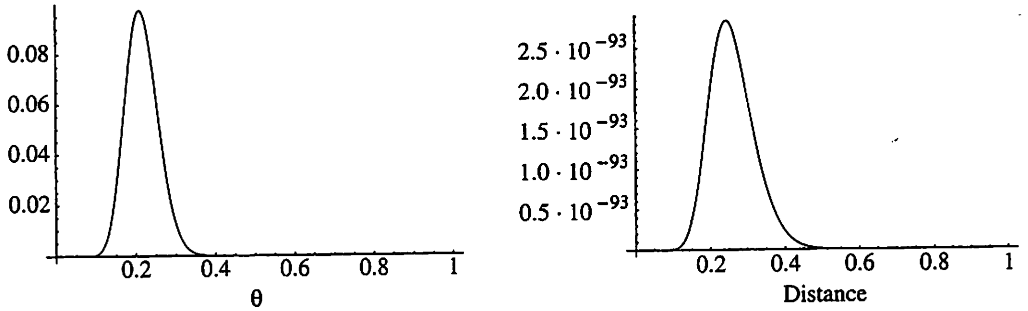

Fig. 6.1 Left: likelihood function of a coin-tossing experiment showing 21 heads and 79 tails. Right: likelihood function of the Jukes–Cantor model of sequence evolution for a sequence with length 100 and 21 observed differences.

that the probability to observe exactly $H = h$ heads in n tosses can be calculated according to the binomial distribution:

$$\Pr[H = h] = \binom{n}{h}\theta^{h}(1-\theta)^{n-h} \tag{6.1}$$

Equation (6.1) can be read in two ways. First, if θ is known, then the probability of $h = 0, \ldots, n$ heads in n tosses can be computed. Second, (6.1) can be seen as a function of θ, where n and h are given; this defines the so-called ***likelihood function***

$$L(\theta) = \Pr[H = h] = \binom{n}{h}\theta^{h}(1-\theta)^{n-h} \tag{6.2}$$

From Fig. 6.1, which illustrates the likelihood function for the coin-tossing example, it can be seen that some hypotheses (i.e. choices of θ) generate the observed data with a higher probability than others. In particular, (6.2) becomes maximal if $\theta = \frac{21}{100}$. This value also can be computed analytically. For ease of computation, first compute the logarithm of the likelihood function, which results in sums rather than products:

$$\log[L(\theta)] = \log\binom{n}{h} + h\log\theta + (n-h)\log(1-\theta) \tag{6.3}$$

The problem is now to find the value of θ $(0 < \theta < 1)$ maximizing the function. From elementary calculus, it is known that relative extrema of a function, $f(x)$, occur at critical points of f, i.e. values x_0 for which either $f'(x_0) = 0$ or $f'(x_0)$ is undefined. Differentiation of (6.3) with respect to θ yields:

$$L'(\theta) = \frac{\partial \log[L(\theta)]}{\partial\theta} = \frac{h}{\theta} - \frac{n-h}{1-\theta} \tag{6.4}$$

This derivative is equal to zero if $\theta_0 = \frac{h}{n}$, positive, i.e. $L'(\theta) > 0$, for $0 < \theta < \theta_0$, and negative for $\theta_0 < \theta < 1$, so that $\log[L(\theta)]$ attains its maximum at $\theta_0 = \frac{h}{n}$. We say $\hat{\theta} = \frac{h}{n}$ is the ***maximum likelihood estimate (MLE)*** of the probability of observing a head in a single coin toss (the hat ˆ notation indicates an estimate rather than the unknown value of θ). In other words, when the value of θ is selected that maximizes (6.3), the observed data are produced with the highest likelihood, which is precisely the ***maximum likelihood (ML) principle.*** However, the resulting likelihoods are usually small (e.g. $L(21/100) \approx 0.0975$); conversely, the likelihoods of competing hypotheses can be compared by computing the odds ratio. Note that the hypothesis that the coin is fair ($\theta = 1/2$) results in a likelihood of $L(1/2) \approx 1.61 \cdot 10^{-9}$; thus, the MLE of $\hat{\theta} = 0.21$ is $6 \cdot 10^7$ times more likely to produce the data than $\theta = 0.5$! This comparison of odds ratios leads to the statistical test procedure discussed in more detail in Chapters 8, 10–12, and 14.

In evolution, point mutations are considered chance events, just like tossing a coin. Therefore, at least in principle, the probability of finding a mutation along one branch in a **phylogenetic tree** can be calculated by using the same maximum-likelihood framework discussed previously. The main idea behind phylogeny inference with maximum-likelihood is to determine the tree topology, branch lengths, and parameters of the evolutionary model (e.g. ***transition/transversion*** ratio, base frequencies, rate variation among sites) (see Chapter 4) that maximize the probability of observing the sequences at hand. In other words, the likelihood function is the conditional probability of the data (i.e. sequences) given a hypothesis (i.e. a model of substitution with a set of parameters θ and the tree τ, including branch lengths):

$$\begin{aligned} L(\tau, \theta) &= \Pr(\text{Data}|\tau, \theta) \\ &= \Pr(\text{aligned sequences}|\text{tree, model of evolution}) \end{aligned} \tag{6.5}$$

The MLEs of τ and θ (named $\hat{\tau}$ and $\hat{\theta}$) are those making the likelihood function as large as possible:

$$\hat{\tau}, \hat{\theta} = \underset{\tau, \theta}{\operatorname{argmax}} L(\tau, \theta) \tag{6.6}$$

Before proceeding to the next section, some cautionary notes are necessary. First, the likelihood function must not be confused with a probability. It is defined in terms of a probability, but it is the probability of the observed event, not of the unknown parameters. The parameters have no probability because they do not depend on chance. Second, the probability of getting the observed data has nothing to do with the probability that the underlying model is correct. For example, if the model states that the sequences evolve according to a tree, although they have recombined, then the final result will still be a single tree that gives rise to the maximum-likelihood value (see also Chapter 15). The probability of the data being given the MLE of the parameters does not provide any hints that the model assumptions are in fact true. One can only compare the maximum-likelihood values with other likelihoods for model parameters that are elements of the model. To determine whether the hypothesis of tree-like evolution is reasonable, the types of relationship allowed among sequences must be enlarged; this is discussed in Chapter 21.

6.2 The formal framework

Before entering the general discussion about maximum-likelihood tree reconstruction, the simplest example (i.e. reconstructing a maximum-likelihood tree for two sequences) is considered. A tree with two **taxa** has only one branch connecting the two sequences; the sole purpose of the exercise is reconstructing the branch length that produces the data with maximal probability.

6.2.1 The simple case: maximum-likelihood tree for two sequences

In what follows, it is assumed that the sequences are evolving according to the Jukes and Cantor model (see Chapter 4). Each position evolves independently from the remaining sites and with the same ***evolutionary rate***. The alignment has length l for the two sequences $S_i = (s_i^1, \ldots, s_i^l)$, $(i = 1, 2)$, where s_i^j is the nucleotide, the amino acid, or any other letter from a finite alphabet at sequence position j in sequence i. The likelihood function is, then, according to (4.31) (Chapter 4):

$$L(d) = \prod_{j=1}^{l} \pi_{s_1^j} P_{s_1^j s_2^j}\left(-\frac{4d}{3}\right) \tag{6.7}$$

where d, the number of substitutions per site, is the parameter of interest and $P_{xy}(t)$ is the probability of observing nucleotide y if nucleotide x was originally present, and $\pi_{s_1^j}$ is the probability of character s_1^j in the equilibrium distribution. From (4.12a) and (4.12b), the following is obtained:

$$P_{xy}\left(-\frac{4}{3}d\right) = \begin{cases} \frac{1}{4}\left(1 + 3\exp\left[-\frac{4}{3}d\right]\right) \equiv \tilde{P}_{xx}(d), & \text{if } x = y \\ \frac{1}{4}\left(1 - \exp\left[-\frac{4}{3}d\right]\right) \equiv \tilde{P}_{xy}(d), & \text{if } x \neq y \end{cases} \tag{6.8}$$

To infer d, the relevant statistic is the number of identical pairs of nucleotides (l_0) and the number of different pairs (l_1), where $l_0 + l_1 = l$. Therefore, the alignment is summarized as $\mathbf{D} = (l_0, l_1)$ and the score is computed as:

$$\log[L(d)] = C + l_0 \log\left[\tilde{P}_{xx}(d)\right] + l_1 \log\left[\tilde{P}_{xy}(d)\right] \tag{6.9}$$

which is maximal if

$$d = -\frac{3}{4}\log\left[1 - \frac{4}{3}\cdot\frac{l_1}{l_1 + l_0}\right]. \tag{6.10}$$

```
L20571   ...AAAGTAATGAAGAAGAACAACAGGAAGTCATGGAGCTTATACATA...
AF10138  ...ATGGAGAAGAAGAAG---------AGACTCTGGCTAAGTTATTGT...
X52154   ...ATGGAGAAGAAGAAG---------AGAGACTGGAACAGCTTATCC...
U09127   ...ATGGGGATAGAGAGGAATTATCCTTGCTGGTGGACATGGGGATT...
U27426   ...AGGGGGATACAGATGAATTGGCAACACTTGTGGAAATGGGGAACT...
U27445   ...AAGGGGATACGGACGAATTGGCAACACTTCTGGAGATGGGGAACT...
U067158  ...AGGGGGACACTGAGGAATTATCAACAATGGTGGATATGGGGCGTC...
U09126   ...GAGGGGATACAGAGGAATTGGAAACAATGGTGGATATGGGGCATC...
U27399   ...AGGGAGATGAGGAGGAATTGTCAGCATTTGTGGGGATGGGGCACC...
U43386   ...AGGGAGATGCAGAGGAATTATCAGCATTTATGGAAATGGGGCATC...
L02317   ...AAGGAGATCAGGAAGAATTATCAGCACTTGTGGAGATGGGGCACC...
AF025763 ...AAGGGGATCAGGAAGAATTGTCAGCACTTGTGGAGATGGGGCATG...
U08443   ...AAGGAGATGAGGAAGCATTGTCAGCACTTATGGAGAGGGGGCACC...
AF042106 ...AAGGGGATCAGGAAGAATTATCGGCACTTGTGGACATGGGGCACC...
```

Fig. 6.2 Part of the mtDNA sequence alignment used as a relevant example throughout the book.

This result is not influenced by the constant C, which only changes the height of the maximum but not its "location." Please note that, the MLE of the number of substitutions per site equals the method-of-moments estimate (see (4.15a)). Therefore, the maximum-likelihood tree relating the sequences S_1 and S_2 is a straight line of length d, with the sequences as endpoints.

This example was analytically solvable because it is the simplest model of sequence evolution and, more importantly, because only two sequences – which can only be related by *one* tree – were considered. The following sections set up the formal framework to study more sequences.

6.2.2 The complex case

When the data set consists of $n > 2$ aligned sequences, rather than computing the probability $P_{xy}(t)$ of observing two nucleotides x and y at a given site in two sequences, the probability of finding a certain column or pattern of nucleotides in the data set is computed. Let D_j denote the nucleotide pattern at site $j \in \{1, \ldots, l\}$ in the alignment (Fig. 6.2). The unknown probability obviously depends on the model of sequence evolution, M, and the tree, τ relating the n sequences with the number of substitutions along each branch of the tree (i.e. the branch lengths). In theory, each site could be assigned its own model of sequence evolution according to the general time reversible model (see Chapter 4) and its own set of branch lengths. Then, however, the goal to reconstruct a tree from an alignment becomes almost computationally intractable and, hence, several simplifications are needed. First, it is assumed that each site s in the alignment evolves according to the same model M; for example, the Tamura–Nei (TN) model (see (4.32a, b, c)) (i.e. γ, κ, and π are assumed the same for each site in the alignment). The assumption also implies that all sites evolve at the same rate μ (see (4.24)). To overcome this simplification, the rate at a site is modified by a rate-specific factor, $\rho_j > 0$.

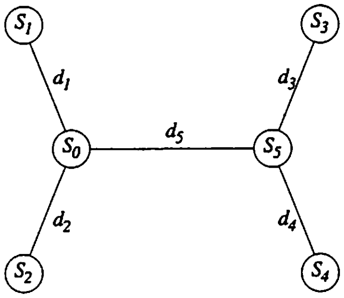

Fig. 6.3 Four-sequence tree, with branch lengths d_1, d_2, d_3, and d_4 leading to sequences S_1, S_2, S_3, and S_4 and branch length d_5 connecting the "ancestral" sequences S_0 and S_5.

Thus, the ingredients for the probability of a certain site pattern are available, and

$$\Pr\left[D_j|\tau, M, \rho_j\right], \quad j = 1, \ldots, l \tag{6.11}$$

specifies the probability to observe pattern D_j. If it is also assumed that each sequence site evolves independently (i.e. according to τ and M, with a site specific rate ρ_j), then the probability of observing the alignment (data) $\mathbf{D} = (D_1, \ldots, D_l)$ equals the product of the probabilities at each site, as follows:

$$L(\tau, M, \rho|\mathbf{D}) \equiv \Pr[\mathbf{D}|\tau, M, \rho] = \prod_{j=1}^{l} \Pr\left[D_j|\tau, M, \rho_j\right] \tag{6.12}$$

When the data are fixed, (6.12) is again a likelihood function (like (6.2) and (6.5)), which allows for the two ways of looking at it (see the previous section). First, for a fixed choice of τ, M, and the site rate vector ρ, the probability to observe the alignment $\mathbf{D}$ can be computed with (6.11). Second, for a given alignment $\mathbf{D}$, (6.12) can be used to find the MLEs.

In what follows, the two issues are treated separately. However, to simplify the matter, it is assumed that the site-specific rate factor ρ_j is drawn from a Γ-**distribution** with expectation 1 and variance $\frac{1}{\alpha}$ (Uzzel & Corbin, 1971; Wakeley, 1993), where α defines the shape of the distribution (see also Section 4.6.1).

6.3 Computing the probability of an alignment for a fixed tree

Consider the tree τ with its branch lengths (i.e. number of substitutions per site), the model of sequence evolution M with its parameters (e.g. transition/transversion ratio, stationary base composition), and the site-specific rate factor $\rho_j = 1$ for each site j. The goal is to compute the probability of observing one of the 4^n possible patterns in an alignment of n sequences. The tree displayed in Fig. 6.3 illustrates the principle for four sequences ($n = 4$). Because the model M is a submodel of the GTR class – that is, a time-reversible model (see Chapter 4) – we can assign any point as a root to the tree for the computation of its likelihood (Pulley Principle, Felsenstein, 1981). Here, we will assume that evolution started from sequence S_0 and then proceeded along the branches of tree τ with branch lengths d_1, d_2, d_3, d_4, and d_5. To compute $\Pr[D_j, \tau, M, 1]$ for a specific site j, where $D_j = (s_1^j, s_2^j, s_3^j, s_4^j)$ are the nucleotides observed, it is necessary to know the ancestral states s_0^j and s_5^j. The conditional probability of the data, given the ancestral states, then will be as follows:

$$\Pr\left[D_j, \tau, M, 1 | s_0^j, s_5^j\right]$$
$$= P_{s_0^j s_1^j}(d_1) \cdot P_{s_0^j s_2^j}(d_2) \cdot P_{s_0^j s_5^j}(d_5) \cdot P_{s_5^j s_3^j}(d_3) \cdot P_{s_5^j s_4^j}(d_4) \qquad (6.13)$$

The computation follows immediately from the considerations in Chapter 4. However, in almost any realistic situation, the ancestral sequences are not available. Therefore, one sums over all possible combinations of ancestral states of nucleotides gaining a so-called *maximum average likelihood* (Steel & Penny, 2000). As discussed in Section 4.4, nucleotide substitution models assume ***stationarity***; that is, the relative frequencies of A, C, G, and T ($\pi_A, \pi_C, \pi_G, \pi_T$) are at equilibrium. Thus, the probability for nucleotide s_0^j will equal its stationary frequency $\pi(s_0^j)$, from which it follows that

$$\Pr\left[D_j, \tau, M, 1\right]$$
$$= \sum_{s_0^j} \sum_{s_5^j} \pi(s_0^j) \cdot P_{s_0^j s_1^j}(d_1) \cdot P_{s_0^j s_2^j}(d_2) \cdot P_{s_0^j s_5^j}(d_5) \cdot P_{s_5^j s_3^j}(d_3) \cdot P_{s_5^j s_4^j}(d_4) \qquad (6.14)$$

Although this equation looks like one needs to compute exponentially many summands, the sum can be efficiently assessed by evaluating the likelihoods moving from the end nodes of the tree to the root (Felsenstein, 1981). In each step, starting from the leaves of the tree, the computations for two nodes are joined and replaced by the joint value at the ancestral node (see Section 6.3.1 for details). This process bears some similarity to the computation of the minimal number of substitutions on a given tree in the ***maximum parsimony*** framework (Fitch, 1971) (see Chapter 8). However, contrary to maximum parsimony, the distance (i.e. number of substitutions) between the two nodes is considered. Under the maximum parsimony framework, if two sequences share the same nucleotide, then the most recent common ancestor also carries this nucleotide (see Chapter 8). In the maximum-likelihood framework, this nucleotide is shared by the ancestor only with a certain probability, which gets smaller if the sequences are only very remotely related.

6.3.1 Felsenstein's pruning algorithm

Equation (6.14) shows how to compute the likelihood of a tree for a given position in a sequence alignment. To generalize this equation for more than four sequences, it is necessary to sum all the possible assignments of nucleotides at the $n-2$ inner nodes of the tree. Unfortunately, this straightforward computation is not feasible, but the amount of computation can be reduced considerably by noticing the following recursive relationship in a tree. Let $D_j = (s_1^j, s_2^j, s_3^j, \ldots, s_n^j)$ be a pattern at a site j, with tree τ and a model M fixed. Nucleotides at inner nodes of the tree are abbreviated as x_i with $i = n+1, \ldots, 2n-2$. For an inner node i with offspring o_1 and o_2, the vector $(\mathbf{L}_j^i = L_j^i(A), L_j^i(C), L_j^i(G), L_j^i(T))$ is defined recursively as

$$L_j^i(s) = \left[\sum_{x\in\{A,C,G,T\}} P_{sx}(d_{o_1}) L_j^{o_1}(x)\right] \cdot \left[\sum_{x\in\{A,C,G,T\}} P_{sx}(d_{o_2}) L_j^{o_2}(x)\right]$$
$$s \in \{A, C, G, T\} \qquad (6.15)$$

and for the leaves

$$L_j^i(s) = \begin{cases} 1, & \text{if } s = s_i^j \\ 0, & \text{otherwise} \end{cases} \qquad (6.16)$$

where d_{o_1} and d_{o_2} are the number of substitutions connecting node i and its descendants in the tree (Fig. 6.4). Without loss of generality, it is assumed that the node $2n-2$ has three offspring: o_1, o_2, and o_3, respectively. For this node, (6.15) is modified accordingly. This equation allows an efficient computation of the likelihood for each alignment position (Fig. 6.4) by realizing that

$$\Pr\left[D_j, \tau, M, 1\right] = \sum_{s\in\{A,C,G,T\}} \pi_s L_j^{2n-2}(s) \qquad (6.17)$$

Equation (6.17) then can be used to compute the likelihood of the full alignment with the aid of (6.12). In practice, the calculation of products is avoided, moving instead to log-likelihoods; that is, (6.12) becomes

$$\begin{aligned}\log\left[L\left(\tau, M, 1\right)\right] &= \log\left[\prod_{j=1}^{l} \Pr\left[D_j, \tau, M, 1\right]\right] \\ &= \sum_{j=1}^{l} \log\left[\Pr\left[D_j, \tau, M, 1\right]\right] \qquad (6.18)\end{aligned}$$

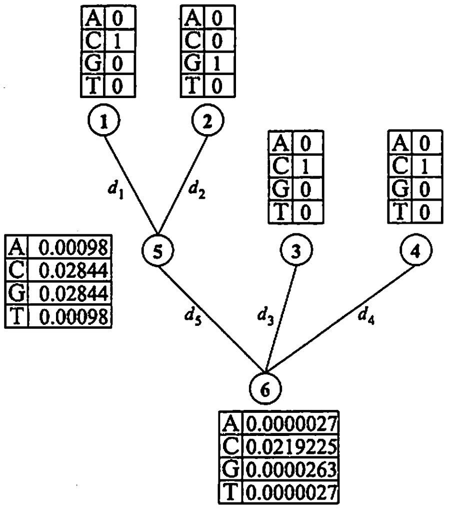

Fig. 6.4 Likelihood computation on a four-taxa tree for an alignment site pattern $D_j = (C, G, C, C)$ with all branch lengths $d_1, \ldots d_5$ set to 0.1. According to (6.8) the probability to observe no mutation after $d = 0.1$ is 0.9058 and for a specific nucleotide pair 0.0314. The values $L^i{}_j(s)$ at the leaves are computed with (6.16), those at the internal nodes with (6.15). For example, to obtain $L^5{}_j(C)$ at node five, (6.15) reduces to $P_{CC}(d_1) \cdot P_{CG}(d_2) = 0.9058 \cdot 0.0314$. The position likelihood according to (6.17) is 0.0054886, the according log-likelihood is −5.2051.

6.4 Finding a maximum-likelihood tree

Equations (6.15) through (6.18) show how to compute the probability of an alignment, if everything were known. In practice, however, branch lengths of the tree are unknown. Branch lengths are computed numerically by maximizing (6.18); that is, by finding those branch lengths for tree τ maximizing the log-likelihood function, which is accomplished by applying numerical routines like Newton–Raphson or Brent's method (Press et al., 1992). Such a computation is usually time-consuming and typically the result depends on the numerical method.

Nevertheless, maximizing the likelihood for a single tree is not the biggest challenge in phylogenetic reconstruction; the daunting task is to actually find the tree among all possible tree structures that maximizes the global likelihood. Unfortunately, for any method that has an explicit optimality criterion (e.g. maximum parsimony, distance methods, and maximum-likelihood), no efficient algorithms are known that guarantee the localization of the best tree(s) in the huge space of all possible tree topologies. The naïve approach to simply compute the maximum-likelihood value for each tree topology is prohibited by the huge number of tree structures, even for moderately sized data sets. The number of (unrooted) binary tree topologies increases tremendously with the number of taxa (n), which can be computed according to

$$t_n = \frac{(2n-5)!}{2^{n-3}(n-3)!} = \prod_{i=1}^{n}(2i-5) \qquad (6.19)$$

When computing the maximum-likelihood tree, the model parameters and branch lengths have to be computed for each tree, and then the tree that yields the highest likelihood is selected. Because of the numerous tree topologies, testing all possible trees is impossible, and it is also computationally not feasible to estimate the model parameters for each tree. Thus, various heuristics are used to suggest reasonable trees, including ***stepwise addition*** (e.g. used in Felsenstein's PHYLIP package: program DNAML, Felsenstein, 1993) and ***star decomposition*** (MOLPHY, Adachi & Hasegawa, 1996) as well as the ***neighbor-joining*** (NJ) algorithm (Saitou & Nei, 1987). Stepwise addition and NJ are discussed in Chapter 8 and Chapter 5, respectively. However, to make this chapter self-consistent we briefly summarize the various heuristics. In our parlance, we are looking for the tree with the highest likelihood. However, the tree rearrangement operations themselves are independent of the objective function.

6.4.1 Early heuristics

Stepwise addition was probably among the first heuristics to search for a maximum-likelihood of a tree. The procedure starts from the unrooted tree topology for three taxa randomly selected from the list of n taxa. Then one reconstructs the corresponding maximum likelihood tree. To extend this tree we randomly pick one of the remaining $n-3$ taxa. This taxon is then inserted into each branch of the best tree. The branch, where the insertion leads to the highest likelihood, will be called insertion branch. Thus, we have a local decision criterion that selects the tree with the highest likelihood from a list of $2k-3$ trees, if k taxa are already in the sub-tree. The resulting tree will then be used to repeat the procedure. After $n-3$ steps, a maximum-likelihood tree is obtained, that is at least locally optimal. That means given the insertion order of the taxa and given the local decision criterion no better tree is possible.

However, we have only computed the maximum-likelihood for $\sum_{i=3}^{n}(2i-5) = (n-2)^2$ trees. Thus, it is possible that another insertion order of the taxa will provide trees with a higher likelihood. To reduce the risk of getting stuck in such local optima, tree-rearrangement operations acting on the full tree were suggested.

6.4.2 Full-tree rearrangement

Full-tree rearrangement operations change the structure of a given tree with n leaves. They employ the following principle. From a starting tree a number of trees (the neighborhood of the starting tree) are generated according to specified rules.

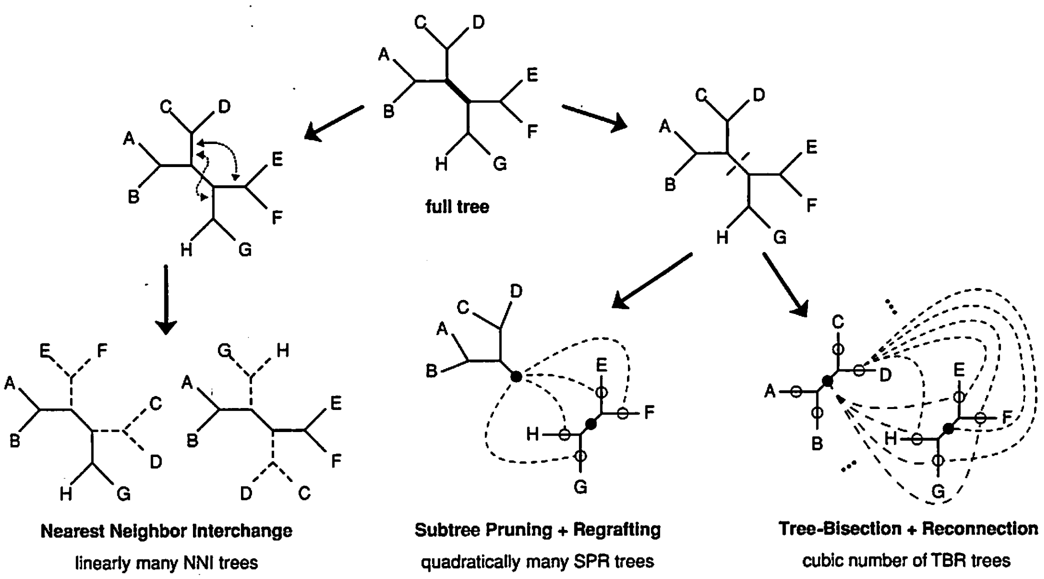
full tree
A
B
C
D
E
F
G
H
Nearest Neighbor Interchange
linearly many NNI trees
Subtree Pruning + Regrafting
quadratically many SPR trees
Tree-Bisection + Reconnection
cubic number of TBR trees

Fig. 6.5 The three basic tree rearrangement operations (NNI, SPR, and TBR) on the thick branch in the full tree. In SPR and TBR all pairs of "circled" branches among the two subtrees will be connected (dashed lines), except the two filled circles to each other, since this yields the full tree again.

For each resulting tree, the maximum-likelihood value is computed. The tree with the highest likelihood is then used to repeat the procedure. The rearrangement typically stops if no better tree is found. This tree is then said to be a locally optimal tree. The chance of actually having determined the globally optimal tree, however, depends on the data and the size of neighborhood.

Three full-tree rearrangement operations are currently popular: ***Nearest neighbor interchange*** (*NNI*), ***sub-tree pruning and regrafting*** (*SPR*) and ***tree-bisection and reconnection*** (*TBR*), confer Fig. 6.5 and see Chapter 8 for more details. Depending on the operation, the size of the neighborhood grows linearly (NNI), quadratically (SPR), or cubically (TBR) with the number of taxa in the full tree.

Different approaches are applied to limit the increase of computation time of the SPR or TBR, while still taking advantage of their extended neighborhood. We will briefly describe some programs and search schemes. Some of these packages have implemented different variants and extensions of the insertion or rearrangement operations. We will not explain them in full detail, but rather refer to the corresponding publications.

6.4.3 DNAml and fastDNAml

The DNAml program (Phylip package, Felsenstein, 1993) and its descendant, fastDNAml (Olsen *et al.*, 1994; Stewart *et al.*, 2001), search by stepwise addition.

Although not turned on by default, the programs allow to apply SPR rearrangements after all sequences have been added to the tree.

Moreover, fastDNAml provides tools to do full tree rearrangements after each insertion step. The user may choose either NNI or SPR, and can also restrict the SPR neighborhood by setting a maximal number of branches to be crossed between pruning and inserting point of the subtree.

6.4.4 PhyML and PhyML-SPR

PhyML (Guindon & Gascuel, 2003) reduces the running time by a mixed strategy. It uses a fast distance based method, BioNJ (Gascuel, 1997), to quickly compute a full initial tree. Then they apply *fastNNI* operations to optimize that tree. During fastNNI all possible NNI trees are evaluated (optimizing only the branch crossed by the NNI) and ranked according to their ML value. Those NNIs which increase the ML value most, but do not interfere with each other, are simultaneously applied to the current tree. Simultaneously applying different NNIs saves time and makes it possible to walk quickly through tree space. On the new current tree fastNNI is repeated until no ML improvement is possible.

Due to their limited range of topological changes NNIs are prone to get stuck in local optima. Hence, a new SPR-based version, PhyML-SPR (Hordijk & Gascuel, 2006), has been devised taking advantage of the larger neighborhood induced by SPR. To compensate for the increased computing time, PhyML-SPR evaluates the SPR neighborhood of the current tree by fast measures like distance-based approaches to determine a ranked list of most promising SPR tree candidates (Hordijk & Gascuel, 2006; for more details). Their likelihood is then assessed by only optimizing the branch lengths on the path from the pruning to the insertion point. If a better tree is found, it takes the status of new current tree.

A fixed number of best candidate trees according to their likelihood are then optimized by adjusting all branch lengths. If now a tree has a higher likelihood than the current one, this tree replaces the old one.

PhyML-SPR allows to alternate SPR and fastNNI-based iterations. Iteration continues until no better tree is found.

6.4.5 IQPNNI

IQPNNI (Vinh & von Haeseler, 2004) uses BioNJ (Gascuel, 1997) to compute the starting tree and fastNNI for likelihood optimization. IQPNNI, however, applies a different strategy to reduce the risk of getting stuck in local optima. When the current tree cannot be improved anymore, IQPNNI randomly removes taxa from the current tree and re-inserts them using a fast quartet-based approach. The new tree is again optimized with fastNNI. If the new tree is better, it then becomes the new starting tree, otherwise the original current tree is kept.

This procedure is either repeated for a user-specified number of iterations or IQPNNI applies a built-in stopping rule, that uses a statistical criterion, to abandon further search (Vinh & von Haeseler, 2004).

Furthermore, IQPNNI provides ML tree reconstruction for various codon models like Goldman & Yang (1994) or Yang & Nielsen (1998). Refer to Chapter 14 for details on such complex models.

6.4.6 RAxML

The RAxML program (Stamatakis, 2006) builds the starting tree based on maximum parsimony (Chapter 8) and optimizes with a variant of SPR called *lazy subtree rearrangement* (LSR, Stamatakis *et al.*, 2005). LSR combines two tricks to reduce the computational demand of SPR operations. First, it assigns a maximal distance between pruning and insertion point for the SPR operations to restrict the size of the neighborhood. The maximal SPR distance (< 25 branches) is determined at the start of the program. Second, LSR optimizes only the branch that originates at the pruning point and the three newly created at the insertion point. The LSRs are repeated many times always using the currently best tree. For the 20 best trees, found during the LSR, the final ML-value is re-optimized by adjusting all branch lengths. The LSR and re-optimization is repeated until no better tree is found.

6.4.7 Simulated annealing

Simulated annealing (Kirkpatrick *et al.*, 1983) is an attempt to find the maximum of complex functions (possibly with multiple peaks), where standard (hill climbing) approaches may get trapped in local optima. One starts with an initial tree, then samples the tree-space by accepting with a reasonable probability a tree with a lower likelihood (down-hill move). Trees with higher likelihood (up-hill moves) are always accepted. This is conceptually related to ***Markov chain Monte Carlo*** (see Chapters 7 and 18). However, as the process continues the down-hill probability is decreased. This decrease is modeled by a so-called cooling schedule. The term "annealing" is borrowed from crystal formation. Initially (high temperature) there is a lot of movement (almost every tree is accepted), then as the temperature is lowered the movements get smaller and smaller. If the decrease in temperature is modeled adequately, then the process will eventually find the ML tree. However, to model the decrease in temperature is not trivial.

First introduced in a parsimony context (Lundy, 1985; Dress & Krüger, 1987), simulated annealing to reconstruct ML trees is applied by SSA (Salter & Pearl, 2001) and RAxML-SA (Stamatakis, 2005). Furthermore, Fleissner *et al.* (2005) use simulated annealing to construct alignments and trees simultaneously.

6.4.8 Genetic algorithms

Genetic algorithms (GA) are an alternative search technique to solve complex optimization problems. They borrow the nomenclature and the optimization decision from evolutionary biology. In fact, GA are a special category of evolutionary algorithms (Bäck & Schwefel, 1993).

The basic ingredients of GA are a population of individuals (in our case a collection of trees) a fitness function (maximum likelihood function according to (6.17)) that determines the offspring number. According to the principles of evolution a tree can mutate (change in branch lengths, NNI, SPR, TBR operations), even trees can exchange sub-trees (recombination). For the mutated tree, the fitness function is computed. The individuals of the next generation are then randomly selected from the mutant trees and the current non-mutated trees according to their fitness (selection step). Typically, one also keeps track of the fittest individual (the tree with the best likelihood). After several generations, evolution stops and the best tree is output.

After having been introduced to phylogenetics in the mid-1990s (e.g. Matsuda, 1995), GARLI (Zwickl, 2006), METAPIGA (Lemmon & Milinkovitch, 2004), and GAML (Lewis, 1998) are examples for applications of GA in phylogenetic inference.

6.5 Branch support

As should be clear by now, none of the above methods guarantee to detect the optimal tree. Hence, biologists usually apply a plethora of methods, and if those reconstruct similar trees one tends to have more confidence in the result.

Typically, tree reconstruction methods are searching for the best tree, leaving the user with a single tree and ML value, but without any estimate of the reliability of its sub-trees.

Several measures are used to assess the certainty of a tree or its branches. The ML values from competing hypotheses can be used in a ***likelihood ratio test*** (*LRT*, see Chapters 10, 11, and 14) or other tests (Chapter 12).

The support of branches are often assessed by employing statistical principles. The most widely used approach to assess branch support seems to be ***bootstrapping*** (Efron, 1979; Felsenstein, 1985), where pseudo-samples are created by randomly drawing with replacement l columns from the original l-column alignment, i.e. a column from the data alignment can occur more than once or not at all in a pseudo-sample. From each pseudo-sample a tree is reconstructed and a consensus tree is constructed, incorporating those branches that occur in the majority of the reconstructed trees. These percentages are used as indicator for the reliability of branches. See Chapter 5 for details on branch support analysis using the bootstrap.

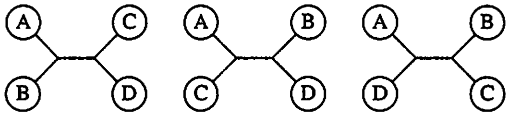

Fig. 6.6 The three different informative tree topologies for the quartet $q = (A, B, C, D)$.

Very similar to the bootstrap is jackknifing, where only a certain percentage of the columns are drawn without replacement (Quenouille, 1956).

Finally, the trees sampled in a Bayesian MCMC analysis are usually summarized in a consensus tree and the tree sample can be used to derive approximate ***posterior probabilities*** for each split or clade (Ronquist & Huelsenbeck, 2003; see also next chapter).

Another method to measure branch support is the ***quartet puzzling*** method implemented in the Tree-Puzzle software, that will be explained in the following section. Although Tree-Puzzle is nowadays not faster than most of the above mentioned ML methods, it is usually faster than running at least 100 bootstraps with an ML method and certainly faster than a Bayesian MCMC analysis.

6.6 The quartet puzzling algorithm

Quartet puzzling (Strimmer & von Haeseler, 1996) utilizes quartets, i.e. groups of four sequences. Quartets are the smallest set of taxa for which more than one unrooted tree topology exists. The three different quartet tree topologies are shown in Fig. 6.6. Quartet-based methods use the advantage that the quartet trees can be quickly evaluated with maximum-likelihood. However, there exist $\binom{n}{4} = \frac{n!}{4!(n-4)!}$ possible quartets in a set of n taxa.

Quartet puzzling is performed in four steps.

6.6.1 Parameter estimation

First Tree-Puzzle estimates the parameters for the evolutionary model. To this end:

(i) The pairwise distance matrix D is estimated for all pairs of sequences in the input alignment and a Neighbor Joining tree is constructed from D.
(ii) Then, maximum-likelihood branch lengths are computed for the NJ topology and parameters of the sequence evolution are estimated.
(iii) Based on these estimates, a new D and NJ tree are computed and Step (ii) is repeated.

Steps (ii) and (iii) are repeated until the estimates of the model parameters are stable.

6.6.2 ML step

To produce the set of tree topologies, the likelihoods of all $3 \times \binom{n}{4}$ quartet tree topologies are evaluated. Then, for each quartet and each topology the corresponding highest likelihood is stored. The algorithm takes into account that two topologies may have similar likelihoods (partly resolved quartet) or that even no topology (unresolved quartet) gains sufficient support (Strimmer *et al.*, 1997).

6.6.3 Puzzling step

Based on the set of supported quartet topologies, trees are constructed by adding taxa in random order. Each taxon is inserted into that branch least contradicted by the set of relevant quartet trees.

This step is repeated many times with different input orders, producing a large set of intermediate trees.

6.6.4 Consensus step

The set of intermediate trees is subsequently summarized by a majority rule consensus tree, the so-called quartet puzzling tree, where the percent occurrences for each branch are considered **puzzle support values.**

6.7 Likelihood-mapping analysis

The chapter so far has discussed the problem of reconstructing a phylogenetic tree and assessing the reliability of its branches. A maximum-likelihood approach may also be used to study the amount of *evolutionary information* contained in a data set. The analysis is based on the maximum-likelihood values for the three possible four taxa trees. If L_1, L_2, and L_3 are the likelihoods of trees T_1, T_2, and T_3, then one computes the posterior probabilities of each tree T_i as $p_i = \frac{L_i}{L_1+L_2+L_3}$. Since the p_i terms sum to 1, the probabilities p_1, p_2, and p_3 can be reported simultaneously as a point P lying inside an equilateral triangle, each corner of the triangle representing one of the three possible tree topologies (Fig. 6.7a). If P is close to one corner – for example, the corner T_1 – the tree T_1 receives the highest support. In a maximum-likelihood analysis, the tree T_i, which satisfies $p_i = \max\{p_1, p_2, p_3\}$, is selected as the MLE. However, this decision is questionable if P is close to the center of the triangle. In that case, the three likelihoods are of similar magnitude; in such situations, a more realistic representation of the data is a star-like tree rather than an artificially *strictly bifurcating tree* (see Section 1.7 in Chapter 1).

Therefore, the ***likelihood-mapping method*** (Strimmer & von Haeseler, 1997) partitions the area of the equilateral triangle into seven regions (Fig. 6.7b). The three trapezoids at the corners represent the areas supporting strictly bifurcating trees (i.e. Areas 1, 2, and 3 in Fig. 6.7b). The three rectangles on the sides represent

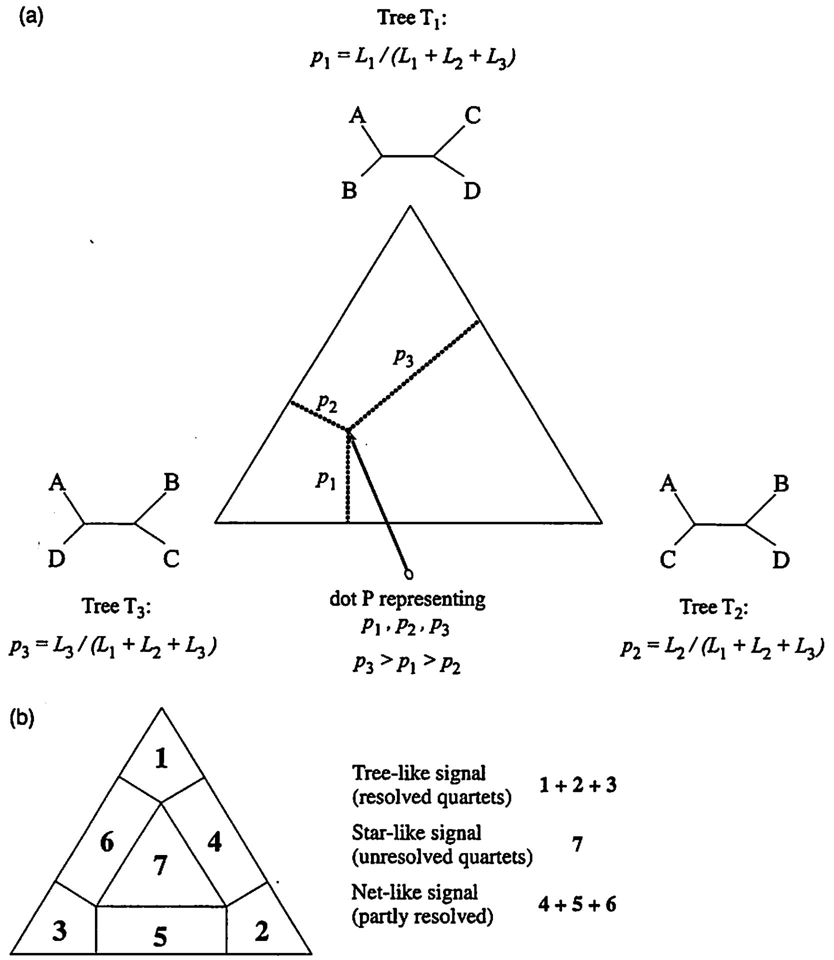
(a)
Tree T_1:
$p_1 = L_1/(L_1 + L_2 + L_3)$
A
C
B
D
p_3
p_2
p_1
A
B
D
C
Tree T_3:
$p_3 = L_3/(L_1 + L_2 + L_3)$
dot P representing
p_1, p_2, p_3
$p_3 > p_1 > p_2$
A
B
C
D
Tree T_2:
$p_2 = L_2/(L_1 + L_2 + L_3)$
(b)
1
6
4
7
3
5
2
Tree-like signal (resolved quartets) 1 + 2 + 3
Star-like signal (unresolved quartets) 7
Net-like signal (partly resolved) 4 + 5 + 6

Fig. 6.7 Likelihood mapping. (a) The three posterior probabilities p_1, p_2, and p_3 for the three possible unrooted trees of four taxa are reported as a point (*P*) inside an equilateral triangle, where each corner represents a specific tree topology with likelihood L_1, L_2, and L_3, respectively. (b) Seven main areas in the triangle supporting different evolutionary information.

regions where the decision between two trees is not obvious (i.e. Areas 4, 5, and 6 in Fig. 6.7b for trees 1 and 2, 2 and 3, and 3 and 1). The center of the triangle represents sets of points *P* where all three trees are equally supported (i.e. Area 7 in Fig. 6.7b). Given a set of *n* aligned sequences, the likelihood-mapping analysis works as follows. The three likelihoods for the three tree topologies of each possible quartet (or of a random sample of the quartets) are reported as a dot in an equilateral triangle like the one in Fig. 6.7a. The distribution of points in the seven areas of the triangle (see Fig. 6.7b) gives an impression of the tree-likeness of the data. Note that, because the method evaluates quartets computed from *n* sequences, which one of the three topologies is supported by any corner of the triangle is not relevant. Only the percentage of points belonging to the areas 1, 2, and 3 is relevant to get an impression about the amount of tree-likeness in the data. To summarize the three corners (Areas 1 + 2 + 3; see Fig. 6.7b) represent fully resolved tree topologies; Area 7 represents star-like phylogenies (Fig. 6.7b); the three Areas 4 + 5 + 6 (see Fig. 6.7b) represent network-like phylogeny, where the data support conflicting tree topologies (see also Chapter 21).

From a biological standpoint, a likelihood mapping analysis showing more than 20%–30% of points in the star-like or network-like area suggests that the data are not reliable for phylogenetic inference. The reasons why an alignment may not be suitable for tree reconstruction are multiple, e.g. noisy data, alignment errors, recombination, etc. In the latter case, methods that explore and display conflicting trees, such as **bootscanning** (see Chapter 16), ***split decomposition*** or NEIGHBOR-NET (see Chapter 21 for network analysis) may give additional information. A more detailed study on quartet mapping is given in Nieselt-Struwe & von Haeseler (2001).