BIOINFORMATICS

Vol. 00 no. 00 2005
Pages 1–9

Improving the Efficiency of SPR Moves in Phylogenetic Tree Search Methods Based on Maximum Likelihood

Wim Hordijk and Olivier Gascuel*

Projet Méthodes et Algorithmes pour la Bioinformatique
LIRMM, UMR CNRS 5506 - Université Montpellier 2
161 rue Ada, 34392 - Montpellier - FRANCE
wim@santafe.edu, gascuel@lirmm.fr

ABSTRACT

Motivation: Maximum likelihood methods have become very popular for constructing phylogenetic trees from sequence data. However, despite noticeable recent progress, with large and difficult data sets (e.g. multiple genes with conflicting signals) current ML programs still require huge computing times and can be trapped in bad local optima of the likelihood function. When this occurs, the resulting trees may still show some of the defects (e.g. long branch attraction) of starting trees obtained using fast distance or parsimony programs.

Methods: Subtree Pruning and Regrafting (SPR) topological rearrangements are usually sufficient to intensively search the tree space. Here, we propose two new methods to make SPR moves more efficient. The first method uses a fast distance-based approach to detect the least promising candidate SPR moves, which are then simply discarded. The second method locally estimates the change in likelihood for any remaining potential SPRs, as opposed to globally evaluating the entire tree for each possible move. These two methods are implemented in a new algorithm with a sophisticated filtering strategy, which efficiently selects potential SPRs and concentrates most of the likelihood computation on the promising moves.

Results: Experiments with real data sets comprising 35 to 250 taxa show that, while indeed greatly reducing the amount of computation, our approach provides likelihood values at least as good as those of the best known ML methods so far, and is very robust to poor starting trees. Furthermore, combining our new SPR algorithm with local moves such as PHYML's nearest neighbor interchanges, the time needed to find good solutions can sometimes be reduced even more.

Availability: Executables of our SPR program and the used data sets are available for download at http://atgc.lirmm.fr/spr.

1 INTRODUCTION

Maximum likelihood (ML) methods have become very popular for constructing phylogenies from sequence data. Felsenstein brought this framework to nucleotide-based phylogenetic inference [8], and it was later also applied to amino acid sequences [14]. Several variants were proposed, among which the widely used Bayesian methods [21, 25, 13]. A number of computer studies [15, 12, 23, 22, 11] have shown that ML programs can recover the correct tree from simulated data sets more frequently than other methods can, which supported numerous observations from real data and explains their popularity.

*to whom correspondence should be addressed

However, the disadvantage of ML methods is that they require much computational effort. Tree inference in the ML setting is computationally hard [5], and all practical approaches rely on heuristics. The main idea behind those heuristics is that the space of possible tree topologies is searched for an optimal topology, optimizing edge lengths along the way. But even computing the optimal values of edge lengths on a single tree is not an easy task. This requires heavy numerical optimization techniques (reviewed in [3]), simply due to the number of parameters ($2n$-3 edges, where n is the number of taxa), and local optima are still possible [4].

Despite these computational difficulties, ML methods have become faster and faster. We distinguish three main components in recent approaches:

(1) *Simultaneous optimization of tree topology and edge lengths.* The first hillclimbing algorithms that were introduced, iterate the following steps: (a) choose a neighboring topology of the current one; (b) optimize the edge lengths of this new topology to obtain its likelihood; (c) if this new topology is better than the current one, then it becomes the new current topology, else the current topology is unchanged. Due to the size of the topology space and the computational intensity of edge length optimization, this approach becomes impractical for even moderate numbers of taxa and limited topological rearrangements. Stochastic approaches, based on MCMC [25, 13], simulated annealing [24, 28] or genetic optimization [17, 16], simultaneously change edge lengths and tree topology, while recent deterministic methods [11, 32, 29] estimate the likelihood of the new topology using approximate edge lengths, which provides lower bounds of its (fully optimized) likelihood. In all cases, likelihood approximations become more and more precise as the algorithm proceeds and as edge lengths become more accurate. This way expensive computations are avoided and the algorithm converges toward local-global optima of the likelihood function.

(2) *Local moves.* Nearest neighbor interchange (NNI) is the simplest topological move: it exchanges two subtrees that are connected by a single edge. Using NNIs, as implemented in PHYML [11], is fast as the change in likelihood for each move can be calculated locally. Moreover, it is shown in [11] that the likelihood improving NNI moves, although evaluated independently, can be performed all at once in most cases. Even though NNI moves do not allow the tree space to be searched intensively, this approach is usually sufficient to greatly improve starting trees, such as those obtained by fast distance-based or parsimony programs, and to get high topological accuracy and satisfactory likelihood values [11].

(3) *Global moves.* Not surprisingly, NNIs can be trapped in (bad)

local optima, and this seems to happen frequently with difficult data sets, e.g. those obtained from multiple genes with conflicting signals. In those cases, it occurs that some of the defects of the starting tree (e.g. long branch attraction) are still present in the resulting tree, even when this does not correspond to the likelihood optimum [20]. In a subtree pruning and regrafting (SPR) move, a subtree is pruned and then regrafted at a different position in the remaining tree. With such moves, getting trapped in bad local optima can often be avoided, resulting in a better and more exhaustive search [9, 31]. However, SPR moves are much more expensive in terms of computation, as the likelihood has to be re-evaluated over the entire tree for each potential move. Recently, [29] showed that the amount of computing time for performing an SPR based search can be reduced by a combination of an efficient implementation of the actual likelihood computations, lazy subtree rearrangements (i.e., pre-evaluating potential moves by some fast likelihood estimate), and carefully ordering potential SPRs, starting from the regraft positions that are close to the original pruned edge, before evaluating unlikely distant positions.

To improve current likelihood-based search methods even more, and most notably to make them less prone to starting tree defects, it is desirable to combine the advantages of both types of moves: the efficiency of local moves and the more elaborate search of global moves. We propose two methods to make SPR moves more efficient. The first method uses a fast distance approach based on the minimum evolution principle [6]. Candidate SPR moves that are unpromising in terms of the minimum evolution criterion are simply discarded, which acts as a first filtering stage. The second method involves updating a limited number of partial likelihoods along the path between the prune and regraft positions (similar to what is done in RAxML [27]), and estimating edge lengths by distance-based analytical formulae; this enables a local approximate (but accurate) calculation of the change in likelihood for candidate SPR moves as opposed to a global (and expensive) computation. This approximate likelihood value is often sufficient to detect truly improving SPRs and, when none are found, is used to filter again potential SPR moves before full likelihood calculations are performed. Our experiments with real data sets indicate that, while indeed greatly reducing the amount of computation, the results of this approach are at least as good as those of the best known ML methods so far, and show excellent robustness to poor starting trees.

The remainder of this paper is organized as follows. The next section will review the basic mathematical notation and equations involved in likelihood computations and the minimum evolution principle on phylogenetic trees, in particular in the context of SPR moves. Section 3 shows how the minimum evolution criterion can be used efficiently to filter out unpromising SPR moves. Section 4 then describes how updating only relevant partial likelihoods on the path between the prune and regraft positions and estimating relevant edge lengths suffices for enabling a fast but accurate estimation of the change in likelihood for an SPR move. Our complete SPR algorithm is presented in section 5, and section 6 shows the results of this algorithm, including the achieved reduction in computation time. Finally, section 7 summarizes the main results and conclusions.

2 NOTATION AND MATHEMATICAL CONCEPTS

Consider a phylogenetic tree as shown in figure 1. The triangles attached to nodes a, b, c, and d represent (possibly empty) subtrees.

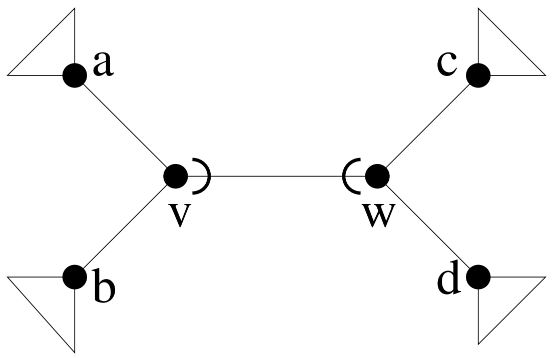

Fig. 1. An example of a phylogenetic tree. Triangles represent (possibly empty) subtrees, and the half-circles on the edge (v, w) represent the two partial likelihoods associated with this edge.

The half-circles on the edge (v, w) connecting nodes v and w represent partial likelihoods. For example, the half-circle next to node v represents the partial likelihood Lk_p of the subtree rooted at node v with child nodes a and b. The likelihood $Lk(T, i)$ of the complete tree T at site i can be calculated locally on the edge (v, w) given these partial likelihoods and the edge length:

$$Lk(T,i) = \sum_{x,y\in\sigma} \pi_x Lk_p(i(v)=x) Lk_p(i(w)=y) P_{xy}(l), \quad (1)$$

where $\sigma = \{A, C, G, T\}$, π_x is the a priori probability of observing nucleotide x, $i(v)$ is the state (nucleotide) at site i at node (taxon) v, and $P_{xy}(l)$ is the probability of a substitution from nucleotide x to y given a time period l, which is the length $d(v, w)$ of edge (v, w). The tree likelihood is then equal to the product of the likelihoods of the sites, as induced by the site independence assumption [8]:

$$Lk(T) = \Pi_i Lk(T, i). \quad (2)$$

Note that equation 1 is given for nucleotide sequences, but of course holds for protein sequences as well by replacing the alphabet σ over which the sum is taken. This equation is also easily adapted to incorporate site-to-site variation using a discrete rate (e.g., gamma) distribution. The full likelihood of a given site is then obtained by summing, over the rate categories, the likelihoods of the site according to each rate, weighted by the probability of each rate category [35]. Finally, every edge defines two partial likelihoods (one in each direction), and all partial likelihoods within the tree can be calculated and updated efficiently (in linear time per site) by performing a pre-order [8] and then a post-order depth-first traversal [1]of the tree, starting from the leaf partial likelihoods that are equal to 0 or 1 depending on the site value for the given taxon.

There are various ways of estimating or optimizing the edge lengths of a given tree. The distance between nodes v and w can be optimized accurately using the partial likelihoods $Lk_p(i(v) = x)$ and $L_p(i(w) = y)$ and equations 1 and 2 above so that the overall likelihood $Lk(T)$ is maximal (an iterative optimization method such as, e.g., Newton-Raphson or Brent [2] can be used for this). Another, much faster but less optimal, way (providing a lower bound of tree likelihood) of estimating $d(v, w)$ is based on the average subtree distance (reviewed in [7]). The *average subtree distance* $\Delta_{v|w}$ between two (non-overlapping) subtrees rooted at nodes v and w, respectively, is the average distance between all taxa in the first subtree and all taxa in the second subtree, and is recursively defined as (referring to figure 1):

$$\Delta_{v|w} = \frac{1}{2}\left(\Delta_{v|c} + \Delta_{v|d}\right) = \frac{1}{2}\left(\Delta_{a|w} + \Delta_{b|w}\right), \tag{3}$$

where $\Delta_{v|w}$ is the actual distance if both v and w are taxa. Note that this definition is the balanced version of average subtree distance, where each subtree has equal weight regardless of the number of taxa it contains. Given the matrix of pairwise evolutionary distances (as obtained by any model, e.g. K2P or HKY), the average subtree distance between all pairs of non-overlapping subtrees can be calculated in $O(n^2)$ time, where n is the total number of taxa [6]. The length of edge (v, w)) can now be estimated as:

$$d(v,w) = \Delta_{v|w} - \frac{1}{2}\Delta_{a|b} - \frac{1}{2}\Delta_{c|d}. \tag{4}$$

An example of an SPR move of (topological) distance s is shown in figure 2. The subtree indicated in bold is pruned at node v_p, and nodes v_0 and v_1 will now be connected by a new edge (indicated by the dashed line between them), while the edges originally connecting node v_p with nodes v_0 and v_1 will be removed. Next, the pruned subtree is regrafted at a distance s from its original position, between nodes v_s and v_{s+1}. This new situation is indicated in blue (both for $s = 1$ and for general values of s). The original edge between nodes v_s and v_{s+1} is removed, and two new edges connecting node v_p with nodes v_s and v_{s+1} are inserted. An SPR move of distance $s = 1$ is, in fact, equivalent to an NNI move.

Since an SPR move changes the topology, and thus the relative subtree positions, many average subtree distances as calculated before the SPR move are not correct anymore. However, in most cases they can be updated in constant time to reflect the new topology, without having to recalculate them all (which would take quadratic time). In particular, two average subtree distances at the regraft position that will be useful later on, $\Delta_{v_s|w_p}$ and $\Delta_{v_{s-1}|w_s}$, can be updated recursively as follows:

$$\Delta^*_{v_s|w_p} = \frac{1}{2}\left(\Delta^*_{v_{s-1}|w_p} + \Delta_{w_s|w_p}\right),$$

with

$$\Delta^*_{v_0|w_p} = \Delta_{v_0|w_p}, \tag{5}$$

and

$$\Delta^*_{v_{s-1}|w_s} = \Delta_{v_{s-1}|w_s} - \left(\frac{1}{2}\right)^s \Delta_{w_p|w_s} + \left(\frac{1}{2}\right)^s \Delta_{v_0|w_s}.$$

Since $\Delta^*_{v_s|w_p}$ is defined recursively in s, in our SPR algorithm (presented in full in section 5) we consider possible regraft positions, given a pruned subtree, recursively in (increasing) distance s.

3 SPR MOVES AND TREE LENGTH

A criterion in phylogenetic inference that is conceptually related to parsimony, and based on average subtree distances, is the *minimum evolution principle*. Several variants of this principle have been proposed, but here we follow the definition and notation of [6, 7]. In particular, we use the concept of *tree length* $L(T)$, which is defined as the sum of the edge lengths of a tree T, and consider the balanced version, first introduced in [19]. The *minimum evolution tree* is then that tree T which minimizes $L(T)$.

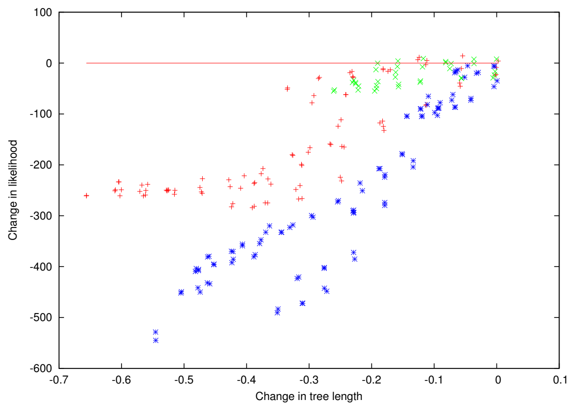

Fig. 3. Correlation between change in tree length and change in likelihood. Each color represents data for one particular pruned subtree and all its candidate regraft positions.

Consider again an SPR move of distance s (as in figure 2). The change in tree length dL_1 resulting from an SPR move of distance $s = 1$ (i.e., an NNI move), is equal to ([6], equation 12):

$$dL_1 = \frac{1}{4}\left[\left(\Delta_{v_0|w_p} + \Delta_{v_2|w_1}\right) - \left(\Delta_{v_0|w_1} + \Delta_{v_2|w_p}\right)\right].$$

From this, a recursive formula dL_s for the change in tree length for an SPR move of distance s can be derived:

$$\begin{aligned} dL_s &= dL_{s-1} + \frac{1}{4}\Big[\left(\Delta^*_{v_{s-1}|w_p} + \Delta_{v_{s+1}|w_s}\right) - \\ &\qquad \left(\Delta^*_{v_{s-1}|w_s} + \Delta_{v_{s+1}|w_p}\right)\Big]. \end{aligned}$$

where the Δ^*'s are as defined in equations 5.

As it turns out, there is a strong correlation between the change in tree length and the change in likelihood for a given SPR move. Figure 3 shows three typical cases. Each collection of dots of one particular color represents data for one particular pruned subtree and all its possible regraft positions (for some given tree). The data shown here (using a real data set of 55 taxa) are representative for this type of data in general, i.e., when considering other trees or data sets as well. Along the horizontal axis the change in tree length (dL_s) is shown, while the vertical axis shows the change in likelihood. Clearly, the regraft positions with the highest change in likelihood for a given pruned subtree also have the highest change in tree length values, and vice versa. Of course, in most cases an arbitrary SPR move will not improve the likelihood of a tree, as for example the collection of blue dots shows: all possible regraft positions result in a reduction in likelihood (negative change). However, if given a pruned subtree there are regraft positions that result in an improvement in likelihood (as with the red and green dots), they always seem to be among the best ones in terms of change in tree length.

Based on this observation, computing changes in likelihood can be avoided altogether for the majority of potential regraft positions. For a given pruned subtree, first the change in tree length dL_s can be calculated quickly for all potential regraft positions (recursively

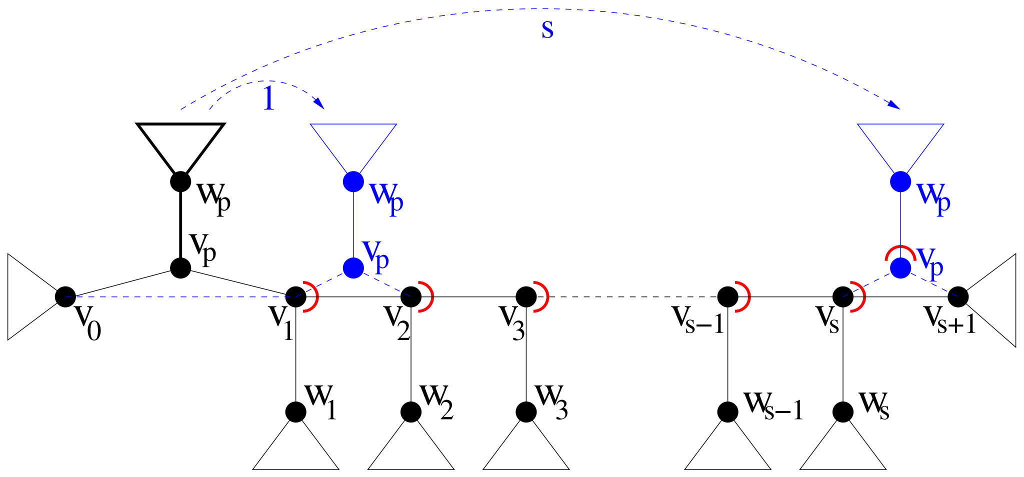

Fig. 2. An example of an SPR move of distance 1 (equivalent to an NNI move) and of distance s in general. The subtree indicated in bold is pruned (at node v_p), and regrafted between nodes v_s and v_{s+1} (indicated in blue). The nodes v_0 and v_1 (at the prune position) will now be connected, indicated by the blue dashed edge.

in distance s), and the actual change in likelihood computations are then done only for the most promising ones. This constitutes our first efficiency improving method.

4 SPR MOVES AND LIKELIHOOD ESTIMATION

One of the main advantages of local moves like NNI is that the tree topology is changed only locally, and the likelihood of the new tree can be calculated efficiently based on equations 1 and 2 (see [11] for details). With SPR moves, on the contrary, the tree topology is often changed significantly, and generally the whole tree has to be re-evaluated (including most or all partial likelihoods). However, to estimate the change in likelihood for a (potential) SPR move to see if it would result in an improved tree, it actually suffices to only update a limited number of relevant partial likelihoods.

To calculate the likelihood of the new tree locally at edge (v_p, w_p) (using equations 1 and 2) after an SPR move, the partial likelihood $Lk_p(i(v_p) = x)$ (represented by the red half-circle on the edge (v_p, w_p) in figure 2) needs to be updated to reflect the new tree topology (note that $Lk_p(i(w_p) = x)$ is not changed by the SPR move). To update $Lk_p(i(v_p) = x)$, however, $Lk_p(i(v_s) = x)$ (represented by the red half-circle on the edge (v_s, v_{s+1})) needs to be updated, and so on, all the way back to node v_1:

$$Lk_p(i(v_k) = x) = \left(\sum_{y\in\sigma} Lk_p(i(v_{k-1}) = y) P_{xy}(d(v_{k-1}, v_k))\right) \times \left(\sum_{z\in\sigma} Lk_p(i(w_k) = z) P_{xz}(d(v_k, w_k))\right). \quad (6)$$

Moreover, w_k is replaced by v_{s+1} and v_k becomes v_s when $k = p$. So, by updating the partial likelihoods on the path between the prune and regraft positions (as indicated by the red half-circles in figure 2), the change in likelihood induced by the SPR move can now be calculated locally. Consequently, for an SPR move of distance s, only $s + 1$ partial likelihoods need to be updated, instead of having to re-evaluate the entire tree. RAxML actually uses a similar approach [26, 29] but in a rooted version, which implies that more partial likelihoods might have to be computed depending on the root position. Furthermore, RAxML stores fewer partial likelihoods (one per edge, instead of two), which is more economical in terms of memory requirements but possibly slower in term of CPU time. Moreover, this method of updating partial likelihoods along the path between prune and regraft positions has (to our knowledge) not yet been described fully and explicitly anywhere else.

Since updating partial likelihoods and estimating changes in tree likelihood depend on edge lengths, accurate estimates for the lengths of the relevant edges at the prune and regraft positions are also needed (while all other edge-lengths are kept constant, at least during the first selection stages, see section 5). An iterative optimization method (as mentioned in section 2) can be used for this, but this is computationally expensive, especially if it has to be performed for each candidate regraft position. Therefore, we use estimates based on the average subtree distances as explained in section 2. Although slightly less accurate than the iterative optimization estimates, they can be calculated much faster.

First consider the new edge connecting nodes v_0 and v_1 at the prune position. A naive and straightforward estimation for the length $d(v_0, v_1)$ of this edge is to simply take the sum of the lengths of the original edges (v_p, v_0) and (v_p, v_1) before pruning, i.e.,

$$d(v_0, v_1) = d(v_p, v_0) + d(v_p, v_1).$$

An estimate based on the average subtree distances is obtained as follows (neglecting taxa within the pruned subtree):

$$d(v_0, v_1) = \Delta_{v_0|v_1} - \frac{1}{2}\Delta_{v_{0a}|v_{0b}} - \frac{1}{2}\Delta_{w_1|v_2},$$

where v_{0a} and v_{0b} are the subtrees rooted at the children of v_0, unless v_0 is a taxon (leaf node), in which case $\Delta_{v_{0a}|v_{0b}} = 0$. Compared to the (more accurate) iterative optimization method, one of these values usually gives an over-estimate, while the other usually gives an under-estimate. So, as a “compromise”, we actually take the average of the two values to get a fairly accurate yet easy to calculate estimate for the length of edge (v_0, v_1).

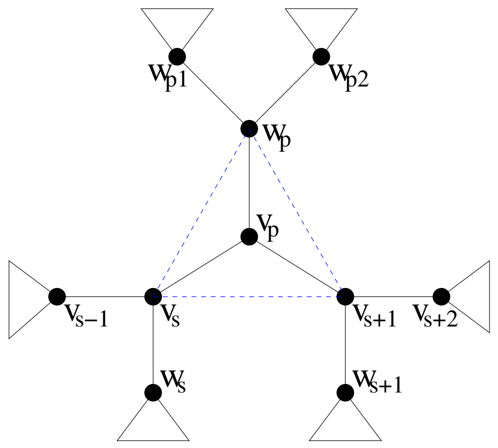

Fig. 4. Local topology at the regraft position of an SPR move of distance s.

Next consider the regraft position. The relevant nodes and edges are shown in figure 4. The edges for which the lengths need to be estimated are (v_p, v_s), (v_p, v_{s+1}), and (v_p, w_p). For $d(v_p, v_s)$ and $d(v_p, v_{s+1})$ we can compute a simple and naive estimate (similar to the one used in RAxML) by taking half of the length of the original edge (v_s, v_{s+1}) (indicated by the dashed blue line between v_s and v_{s+1}). The length of edge (v_p, w_p) can then be estimated as

$$d(v_p, w_p) = d(v_{s+1}, w_p) - d(v_p, v_{s+1})$$

The calculation of $d(v_{s+1}, w_p)$ (indicated by the dashed blue line between v_{s+1} and w_p) is given below.

Estimates based on the average subtree distances are slightly more complicated than at the prune position, but can be calculated from estimates for the lengths of the dashed blue "edges" in figure 4, i.e., $d(v_s, w_p)$, $d(v_{s+1}, w_p)$, and $d(v_s, v_{s+1})$. For the third one, $d(v_s, v_{s+1})$, we can take the length of the original edge (v_s, v_{s+1}). The distance between nodes v_{s+1} and w_p can be directly estimated as:

$$d(v_{s+1}, w_p) = \Delta_{v_{s+1}|w_p} - \frac{1}{2}\Delta_{v_{s+2}|w_{s+1}} - \frac{1}{2}\Delta_{w_{p1}|w_{p2}}.$$

Finally, the distance between nodes v_s and w_p can be calculated similarly as:

$$d(v_s, w_p) = \Delta^*_{v_s|w_p} - \frac{1}{2}\Delta^*_{v_{s-1}|w_s} - \frac{1}{2}\Delta_{w_{p1}|w_{p2}},$$

where the Δ^*'s are again as defined in section 2, and $\Delta_{w_{p1}|w_{p2}} = 0$ if w_p is a leaf.

Once these three distances are calculated, the required edge length estimates are simply

$$\begin{aligned}
d(v_p, w_p) &= \frac{1}{2}[\quad d(v_s, w_p) + d(v_{s+1}, w_p) - d(v_s, v_{s+1})],\\
d(v_p, v_s) &= \frac{1}{2}[\quad d(v_s, w_p) - d(v_{s+1}, w_p) + d(v_s, v_{s+1})],\\
d(v_p, v_{s+1}) &= \frac{1}{2}[-d(v_s, w_p) + d(v_{s+1}, w_p) + d(v_s, v_{s+1})].
\end{aligned}$$

At the regraft position, as at the prune position, the naive estimates and the average subtree distance based estimates also tend to give over- and under-estimates compared to the more accurate iterative edge length optimization. So here too, we take the average of both estimates for each of the three relevant edges.

With these edge length estimates, and the updated partial likelihoods along the path between the prune and regraft position, an estimate for the change in likelihood of a given SPR move can be obtained very quickly. This results in an underestimate, but in practice it turns out that it is usually not too far off from the actual change in likelihood (when edge lengths are fully optimized). Moreover, this estimation method avoids a significant amount of computation, and constitutes our second efficiency improving method.

5 THE SPR ALGORITHM

The main idea of our SPR algorithm is to consider candidate SPR moves, accepting those that result in an improvement in likelihood, but using the methods introduced in the previous sections to discard unpromising moves and to reduce the amount of computation necessary for those moves that are not already filtered out at the first stage. Briefly, each subtree t in the current tree T is considered for pruning in turn. Given a pruned subtree t, all potential regraft positions for t are considered recursively in increasing distance s. The change in tree length dL_s is calculated for each candidate position (as explained in section 3), and a list `rgrft_cand` of the `N_RGRFT` best ones is maintained and updated along the way.

Next, for each candidate in the `rgrft_cand` list (starting with the best one in terms of the dL_s value), the relevant edge lengths are estimated (as explained in the previous section) and the partial likelihoods along the path from the prune to the regraft positions are updated. The change in likelihood is then estimated (locally using equations 1 and 2), and as soon as an improvement is found, this candidate regraft position is accepted and the remaining ones are discarded. This procedure is then repeated for the next subtree t (but now on a possibly improved and updated tree T). So, for each next pruned subtree t, a new list `rgrft_cand` of regraft position candidates is constructed.

During the estimation of the change in likelihood for the various candidate SPR moves, a second list `optim_cand` is maintained and updated with the `N_OPTIM` best candidate moves in terms of these changes in likelihood. Thus, this second list is constructed over *all* possible pruned subtrees t and their respective potential regraft positions. If improvements were already found during the previous stage (i.e., calculating dL_s values and estimating changes in likelihood), then the algorithm will exit and return `true`, indicating that the tree was improved, and this second list is ignored. However, if no improvement was found at all during the previous stage, the `optim_cand` list (which now obviously contains only candidates with negative changes in likelihood) is considered in the following way. For each candidate in the list (starting with the best, although negative, one), perform the SPR move as indicated by this candidate, and use an iterative method (Brent optimization in our implementation) to optimize the lengths of the three relevant edges at the regraft position, and calculate the new likelihood of the tree. If this results in an improvement, accept the move, discard the remaining candidates, exit and return `true`. Otherwise, try the next candidate in the list. The idea behind this procedure is that an iterative optimization method, although computationally expensive, usually gives slightly better estimates for the relevant edge lengths than the fast average subtree distance based estimates. It can, and does, happen that with these more accurate edge length estimates the change in likelihood of one of these N_OPTIM candidates does indeed become positive, and thus give rise to an improved tree after all.

Finally, if this "local optimization" procedure does not result in any improvements either, the best N_GLOBL candidates from the optim_cand list are considered for "global optimization". For each candidate in the list (again starting with the best one based on the local optimizations from the previous stage), perform the SPR move as indicated by this candidate, use an iterative method to optimize the lengths of *all* the edges in the tree, and calculate the new likelihood of the tree. If this results in an improvement, accept the move, discard the remaining candidates, exit and return true. Otherwise, try the next candidate in the list. Here, again, the idea is that an improvement might be found by doing an even more elaborate and complete edge length optimization. However, since this global edge length optimization is computationally very expensive, it is only done for the N_GLOBL best candidates, where usually N_GLOBL << N_OPTIM. If this last procedure does not result in an improvement, then the algorithm will exit and return false, indicating no improvements could be found. So, at each next stage of the algorithm, fewer candidates are considered, while the corresponding change in likelihood estimations become more accurate but also require more intensive computation.

To perform an actual tree search, the algorithm can be performed repeatedly until it returns false. Or, alternatively, it can be combined with a local search method such as NNI, where rounds of NNI moves and SPR moves are alternated, for example. The complete SPR algorithm as described above is presented more formally next.

SPR ()

```
1. Calculate the average subtree distances for all pairs of (non-
   overlapping) subtrees.
2. For each next subtree $t$ do:
   a. Clear the rgrft_cand list.
   b. Prune ($t$) and estimate $d(v_0, v_1)$.
   c. Recursively consider potential regraft positions at increasing
      distance $s$. At each recursion step do:
      (1) Calculate the relevant $\Delta^*$ values and compute $dL_s$.
      (2) If $dL_s$ is within the N_RGRFT best ones so far, store the
          current regraft position together with its $dL_s$ value, the
          current path from the prune position, and the $\Delta^*$ values in
          the rgrft_cand list, sorted in decreasing $dL_s$ values.
   d. For each candidate in the rgrft_cand list (starting at the
      first, or best, one) do:
      (1) Update the partial likelihoods $Lk_p$ along the path between
          the prune and candidate regraft positions.
      (2) Estimate edge lengths at the candidate regraft position
          using the average subtree distances and stored $\Delta^*$ values.
      (3) Regraft $t$ at the candidate regraft position, set the relevant
          edge lengths, update the $Lk_p(i(v_p) = x)$'s using eqn. 6,
          and calculate the likelihood $Lk(T)$ using eqn.'s 1 and 2.
      (4) If the new $Lk(T)$ value is an improvement, accept the cur-
          rent move, update the complete tree (partial likelihoods,
          edge lengths, average subtree distances, etc.), discard the
          remaining candidate regraft positions and go back to step
          2 to try the next subtree.
      (5) If no improvement, undo the regraft operation and reset
          the edge lengths and partial likelihoods. If the change in
          likelihood (although negative) is within the N_OPTIM best
          ones so far, store the current prune and regraft positions
          together with other relevant data (path, $\Delta^*$'s, etc.) in the
          optim_cand list, sorted in decreasing order.
      (6) Go to step 2(e) and try the next candidate.
   e. Go back to step 2 to try the next subtree.
3. If an improvement was found in step 2, exit and return true.
4. Otherwise, for each candidate in the optim_cand list (starting
   at the first, or best, one) do:
   a. Perform the SPR move as indicated by the current candidate
      and update the partial likelihoods along the path between the
      prune and regraft positions.
   b. Optimize the relevant edge lengths at the regraft position
      using an iterative optimization method.
   c. Calculate the new likelihood and if it results in an improve-
      ment, accept the current move, update the complete tree, exit
      and return true.
   d. If no improvement, undo the move and go back to step 4 to
      try the next candidate.
5. For the best N_GLOBL candidates in the optim_cand list
   do:
   a. Perform the SPR move as indicated by the current candidate.
   b. Update all partial likelihoods and perform global edge length
      optimization using an iterative optimization method.
   c. If an improvement in likelihood results, accept the current
      move, exit and return true.
   d. If no improvement, undo the move and go back to step 5 to
      try the next candidate.
6. Exit and return false.
```

Note that there are various parameters in the SPR algorithm, in particular N_RGRFT, N_OPTIM, and N_GLOBL. Of course the performance of the algorithm depends largely on the chosen values for these parameters. Currently it is not obvious what the best values are, as this in turn depends on the data sets considered, and they will have to be found by trial and error. The next section will show the best results we have obtained so far, but more investigation needs to be done in terms of finding good parameter values, or better yet, a systematic way to set them depending on the given data.

6 RESULTS

In this section, we show results of applying our SPR algorithm to some real data sets from the benchmark set used in [29]. In particular, data sets with a number of taxa of 101 (101_SC), 150 (150_ARB), and 250 (250_ARB) were considered. Additionally, we considered the four data sets that were provided with publications in a recent issue of Systematic Biology. These data sets contain 132 [33], 42 [36], 39 [18], and 35 [34] taxa, respectively.

Our SPR algorithm has been implemented in the PHYML program [11], in such a way that it can be used either by itself or in combination with the NNI algorithm provided by PHYML. In the latter case, the SPR algorithm is called once each time the NNI moves are stuck on a local optimum. If the SPR algorithm is able

Table 1. Search results of the different programs on the given data sets. Results for random (rnd) and parsimony (prs) trees are averages over 10 trees each. The ”best” score is the best likelihood value found over all (20) starting trees.

#taxa	start	SPR	PHYML+SPR	RAxML
101	rnd/prs	-73869/-73874	-73878/-73869	-73926/-73870
	best	-73862	-73862	-73862
150	rnd/prs	-76856/-76856	-76855/-76858	-77161/-76853
	best	-76849	-76849	-76849
250	rnd/prs	-130920/-130940	-130902/-130920	-131325/-130897
	best	-130894	-130891	-130889
132	rnd/prs	-50496/-50539	-50502/-50539	-50529/-50530
	best	-50477	-50481	-50477
42	rnd/prs	-5732/-5734	-5734/-5734	-5743/-5734
	best	-5731	-5731	-5731
39	rnd/prs	-2947/-2946	-2946/-2946	-2953/-2947
	best	-2945	-2945	-2945
35	rnd/prs	-1084/-1084	-1084/-1084	-1087/-1084
	best	-1084	-1084	-1084

to improve the tree and get out of the local optimum, the NNI algorithm is applied again until it is trapped in another local optimum, etc. If the SPR algorithm cannot find an improvement anymore either, the program terminates.

We applied both versions (SPR and PHYML+SPR) on the mentioned data sets, and used both random trees and maximum parsimony trees as starting points. Random trees are useful to check that the algorithm is not affected by potentially poor starting trees, while starting with parsimony trees corresponds to standard use, notably regarding run time. The maximum parsimony trees were created using RAxML [29], and the results of our algorithms are compared to this program. The HKY model of nucleotide substitution was used in all cases, and the transition/transversion parameter is estimated by all programs during the search.

Table 1 shows the likelihood values found by the different programs. The results for random trees and parsimony trees are averages over 10 trees each. The rows labeled ”best” shows the best likelihood score found by each program over all 20 starting trees (10 random + 10 parsimony). Since there is a difference in the way likelihood values are calculated between PHYML and RAxML, all final trees found by RAxML were re-evaluated using PHYML (without doing any further optimization, but just evaluating the likelihood of the given tree) to enable a direct comparison.

As the table shows, there is not one program that is *always* the best, but they are all able to find the same best solution (although not necessarily on the same starting tree). The most striking difference is that SPR and PHYML+SPR are more robust with poor (random) starting trees than RAxML, and should thus be more resistant to artifacts (e.g. long branch attraction) that sometimes affect the fast (distance-based or parsimony) methods being used to build initial trees. For example, with 250 taxa SPR and PHYML+SPR are on average about 400 log-likelihood points better than RAxML, which is considerable. With less taxa, the difference is lower but still quite significant. Surprisingly, SPR and PHYML+SPR even seem to perform somewhat better (on average) on random trees than on parsimony trees, but more results are needed to confirm this tendency.

Table 2. Average running times of the three programs on the random (rnd) and parsimony (prs) starting trees.

#taxa	start	SPR	PHYML+SPR	RAxML
101	rnd/prs	7:33/6:45	9:50/6:00	23:52/15:30
150	rnd/prs	15:29/12:50	21:00/13:00	36:00/14:30
250	rnd/prs	2:10:00/2:00:00	1:25:00/1:21:00	2:16:00/0:51:44
132	rnd/prs	14:39/16:00	10:19/5:13	20:56/9:31
42	rnd/prs	0:13/0:10	0:16/0:06	0:15/0:11
39	rnd/prs	0:06/0:05	1:03/0:04	0:12/0:10
35	rnd/prs	0:02/0:02	0:02/0:01	0:04/0:04

Table 2 shows the running times of the different programs on the random and parsimony trees (again averaged over the 10 trees). The runs were performed on an unloaded linux PC with an Intel Pentium 4 (3.0GHz) processor and 512MB of RAM. As the table shows, the running times are of the same order of magnitude for the different programs. However, a direct comparison of running times between these two programs (although useful for practical purposes), does not give much insight into the improvements in efficiency that is achieved by our proposed two methods. For example, the SPR algorithm in RAxML has a very efficient but also very specific (nucleotide data only) implementation for storing trees and calculating likelihoods [30, 27], whereas PHYML (and thus our SPR implementation) is more generic (both nucleotide and protein data) and consequently less optimized for speed. So, the efficieny of RAxML comes to a large extent from implementation optimizations, whereas here we focus more on algorithmic aspects. In the end, of course, it would pay off to combine both approaches.

Considering the results as presented in tables 1 and 2, it can be concluded that the search performance of our algorithm is at least comparable to that of RAxML (which produced the best-known trees to date on a benchmark set of real alignments [29]) with parsimony starting trees, and better with (poor) random starting trees. However, the results do depend on the parameter settings that were used. The results shown here were the best ones obtained after trying a (limited) number of different parameter settings. In all cases we used `N_OPTIM`=100, and for `N_RGRFT` and `N_GLOBL` it seems that values of about 15-20% and 10% (respectively) of the number of edges in the tree give the best results. In practice, we also used a cut-off `MAX_DIST` for the potential regraft positions considered: regraft positions more than a distance `MAX_DIST` from the prune position were not evaluated at all. This parameter (also used by RAxML, although slightly differently) provides another way of reducing the total number of likelihood computations performed by the algorithm. In our experiments this parameter was set to 10% of the number of edges, which is a relatively high value regarding the diameter of phylogenies [10] and resulted in a low influence on the output trees. However, as already mentioned, more tests need to be done to find optimal parameter settings, or some systematic way of setting these values.

Having verified that our SPR algorithm can live up to the best known performance so far, of course the main question is how much the efficiency of the SPR search has been improved by our two introduced methods. Table 3 shows to what extent likelihood computations have been avoided and reduced in our experiments. The second column (a) in this table shows the (average) number of dL_s

Table 3. The reduction in likelihood computations achieved by our algorithm. The columns show the number of candidate SPR moves for which: (a) the change in tree length is calculated, (b) the change in likelihood is estimated, (c) local edge length optimization (at the regraft position) is performed to estimate the change in likelihood, and (d) global edge length optimization is performed.

# taxa	(a) tree length	(b) likelihood	(c) local	(d) global
101	402676	92989	256	22
150	844105	142422	279	31
250	4519332	979891	662	63
132	1100011	242133	792	45
42	28185	8932	237	10
39	28382	12619	291	10
35	22776	9878	291	6

(change in tree length) computations performed during a search of our SPR algorithm, averaged over the 10 parsimony starting trees (results on random trees are similar). The third column (b) shows the number of actual change in likelihood estimations that were computed by only considering the N_RGRFT best candidate regraft positions for each pruned subtree. The fourth column (c) shows the number of times local edge length optimization (at the regraft position) has been performed if no improvements could be found with the analytical estimates. Finally, the fifth column (d) shows the number of times global edge length optimization was performed if no improvements in likelihood could be found with local optimization. As the table clearly shows, our progressive filtering method has an enormous impact.

To measure this impact on the actual running time of the SPR algorithm, we compared the following variants of the algorithm on two data sets of 55 and 101 taxa, respectively:

1. The "regular" algorithm as presented in section 5, with N_RGRFT=15 (about 15% of the number of edges) for the 55-taxon data set, and N_RGRFT=40 (20% of the number of edges) for the 101-taxon data set.
2. As variant 1, but with N_RGRFT=100% of the number of edges for both data sets. In other words, this variant estimates the change in likelihood for *all* candidate SPR moves considered.
3. As variant 1, but performing global likelihood computations (i.e., updating and re-evaluating the entire tree) instead of updating only relevant partial likelihoods and performing a local likelihood computation.
4. A combination of variants 2 and 3, i.e., setting N_RGRFT=100% and performing global likelihood computations.

As starting trees we used the BIONJ tree as computed by PHYML for the 55-taxon data set and one of the 10 parsimony trees used above for the 101-taxon data set. All four variants of the algorithm found the best known likelihood values for these data sets (-24583 and -73862, respectively), and the times necessary to find these solutions are presented in table 4.

As the table clearly shows, the reduction in runtime is very significant for both methods, while the algorithm is still able to find the best known solutions. Avoiding likelihood computations by only estimating likelihood changes for the most promising candidate moves based on the change in tree length (comparing variant 3 with variant 4) results in a decrease in runtime by a factor of about 3. Reducing the amount of computation by estimating the change in likelihood locally after updating only relevant partial likelihoods and quickly estimating edge lengths (comparing variant 2 with variant 4) improves the runtime by a factor of about 10 to 20. Combining the two methods together (comparing variant 1 with variant 4) gives a factor of 36 and 60, respectively, in runtime improvement. Finally, combining the SPR algorithm with local (NNI) moves can sometimes provide even more of a speed-up. For example, on the 101-taxon data set and the same parsimony starting tree, the combination of PHYML+SPR found the best solution in just under 5 minutes.

Table 4. The running times for the four variants of the SPR algorithm on two data sets of 55 and 101 taxa, respectively.

# taxa	55		101	
variant	running time	factor	running time	factor
1	0:00:26	36.0	0:07:42	60.3
2	0:01:13	12.8	0:30:13	21.6
3	0:05:06	3.1	2:46:15	2.8
4	0:15:36	1.0	7:44:21	1.0

7 CONCLUSIONS AND FURTHER WORK

We have introduced two methods that can be used to drastically reduce or even avoid expensive likelihood computations in SPR-based search algorithms on phylogenetic trees using maximum likelihood. The first method avoids likelihood computations altogether by only calculating changes in likelihood for the most promising SPR moves in terms of the change in tree length, a statistic based on the minimum evolution principle that can be calculated efficiently. The second method reduces the amount of (remaining) likelihood computation by updating only relevant partial likelihoods, estimating relevant edge lengths using analytical formulae, and then calculating the change in likelihood for candidate SPR moves locally, instead of having to re-evaluate the entire tree. Our results show that this indeed gives rise to a significant reduction in both the number of likelihood computations and the running time of the algorithm, while maintaining a search performance comparable to that of the best known methods to date.

It is expected that even better results can be achieved by investigating more thoroughly the various parameters that are involved in the algorithm. The speed of the algorithm can possibly still be improved by finding optimal parameter values, or better yet, a systematic way of setting these values based on the input data. Moreover, combining our (algorithmic) methods with a highly optimized implementation for performing likelihood computations (such as RAxML) will most likely result in an extremely efficient program for performing SPR searches on phylogenetic trees using maximum likelihood.

ACKNOWLEDGMENTS

We thank the ACI IMPBIO (French Ministry of Research) for financial support for this project, Stéphane Guindon for comments and help with PHYML, and Alexandros Stamatakis for comments and details on the RAxML implementation.

REFERENCES

[1]V. Berry and O. Gascuel. Inferring evolutionary trees with strong combinatorial evidence. *Theoretical Computer Science*, 240(2):271–298, 2000.

[2]R. Brent. *Algorithms for minimization without derivatives*. Prentice-Hall, 1973.

[3]D. Bryant, N. Galtier, and M.-A. Poursat. Likelihood calculations in phylogenetics. In O. Gascuel, editor, *Mathematics of Evolution and Phylogeny*, pages 33–62. Oxford University Press, 2005.

[4]B. Chor, M. Hendy, B. Holland, and D. Penny. Multiple maxima of likelihood in phylogenetic trees: An analytic approach. *Molecular Biology and Evolution*, 17:1529–1541, 2000.

[5]B. Chor and T. Tuller. Maximum likelihood of evolutionary trees is hard. In *RECOMB05*, 2005.

[6]R. Desper and O. Gascuel. Fast and accurate phylogeny reconstruction algorithms based on the minimum-evolution principle. *Journal of Computational Biology*, 9(5):687–705, 2002.

[7]R. Desper and O. Gascuel. The minimum evolution distance-based approach to phyologenetic inference. In O. Gascuel, editor, *Mathematics of Evolution and Phylogeny*, pages 1–32. Oxford University Press, 2005.

[8]J. Felsenstein. Evolutionary trees from DNA sequences: A maximum likelihood approach. *Journal of Molecular Evolution*, 17:368–376, 1981.

[9]J. Felsenstein. PHYLIP—phylogeny inference package (version 3.2). *Cladistics*, 5:164–166, 1989.

[10]O. Gascuel. Evidence for a relationship between algorithmic scheme and shape of inferred trees. In W. Gaul, O. Opitz, and M. Schader, editors, *Data Analysis, Scientific Modeling and Practical Applications*, pages 157–168. Springer, 2000.

[11]S. Guindon and O. Gascuel. A simple, fast, and accurate algorithm to estimate large phylogenies by maximum likelihood. *Systematic Biology*, 52(5):696–704, 2003.

[12]J. P. Huelsenbeck. Performance of phylogenetic methods in simulation. *Systematic Biology*, 44:1748, 1995.

[13]J. P. Huelsenbeck and F. Ronquist. Mrbayes: Bayesian inference of phylogeny. *Bioinformatics*, 17:754–755, 2001.

[14]H. Kishino, T. Miyata, and M. Hasegawa. Maximum likelihood inference of protein phylogeny and the origin of chloroplasts. *Journal of Molecular Evolution*, 31:151–160, 1990.

[15]M. K. Kuhner and J. Felsenstein. A simulation comparison of phylogeny algorithms under equal and unequal evolutionary rates. *Molecular Biology and Evolution*, 11:459–468, 1994.

[16]A. Lemmon and M. Milinkovitch. The metapopulation genetic algorithm: An efficient solution for the problem of large phylogeny estimation. *Proceedings of the National Academy of Science*, 99:10516–10521, 2002.

[17]P. Lewis. A genetic algorithm for maximum likelihood phylogeny inference using nucleotide sequence data. *Molecular Biology and Evolution*, 15:277–283, 1998.

[18]K. G. McCracken and M. D. Sorenson. Is homoplasy or lineage sorting the source of incongruent mtdna and nuclear gene trees in the stiff-tailed ducks (nomonyx-oxyura). *Systematic Biology*, 54(1), 2005.

[19]Y. Pauplin. Direct calculation of a tree length using a distance matrix. *Journal of Molecular Evolution*, 51:41–47, 2000.

[20]H. Philippe and J. Pons. personal communication.

[21]B. Rannala and Z. Yang. Probability distribution of molecular evolutionary trees: A new method of phylogenetic inference. *Journal of Molecular Evolution*, 43:304–311, 1996.

[22]V. Ranwez and O. Gascuel. Improvement of distance-based phylogenetic methods by a local maximum likelihood approach using triplets. *Molecular Biology and Evolution*, 19:1952–1963, 2002.

[23]M. Rosenberg and S. Kumar. Traditional phylogenetic reconstruction methods reconstruct shallow and deep evolutionary relationship equally well. *Molecular Biology and Evolution*, 19:1823–1827, 2001.

[24]L. Salter and D. Pearl. Stochastic search strategy for estimation of maximum likelihood phylogenetic trees. *Systematic Biology*, 50:717, 2001.

[25]D. Simon and B. Larget. Bayesian analysis in molecular biology and evolution (BAMBE), version 2.03beta, 2000.

[26]A. Stamatakis. *Distributed and Parallel Algorithms and Systems for Inference of Huge Phylogenetic Trees based on the Maximum Likelihood Method*. PhD thesis, Technische Universität München, Germany, 2004.

[27]A. Stamatakis, 2005. personal communication.

[28]A. Stamatakis. An efficient program for phylogenetic inference using simulated annealing. In *Proceedings of 19th IEEE/ACM International Parallel and Distributed Processing Symposium*, 2005.

[29]A. Stamatakis, T. Ludwig, and H. Meier. RAxML-III: A fast program for maximum likelihood-based inference of large phylogenetic trees. *Bioinformatics*, 21(4):456–463, 2005.

[30]A. Stamatakis, T. Ludwig, H. Meier, and M. J. Wolf. AxML: a fast program for sequential and parallel phylogenetic tree calculations based on the maximum likelihood method. In *Proceedings of 1st IEEE Computer Society Bioinformatics Conference (CSB2002)*, pages 21–28, 2002.

[31]D. Swofford. *PAUP*—Phylogenetic Analysis Using Parsimony (*and other methods)*. Sinauer, Sunderland, 1996.

[32]S. Vinh and A. von Haeseler. IQPNNI: moving fast through tree space and stopping in time. *Molecular Biology and Evolution*, 21(8):1565–1571, 2004.

[33]A. P. Vogler, A. Cardoso, and T. G. Barraclough. Exploring rate variation among and within sites in a densely sampled species tree: species level phylogenetics of north american tiger beetles (genus cicindela). *Systematic Biology*, 54(1), 2005.

[34]R. C. Winkworeth, D. Bryant, P. J. Lockhart, D. Havell, and V. Moulton. Biogeographic interpretation of splits graphs: least squares optimisation of branch lengths. *Systematic Biology*, 54(1), 2005.

[35]Z. Yang. Maximum likelihood phylogenetic estimation from DNA sequences with variable rates over sites: Approximate methods. *Journal of Molecular Evolution*, 39:306–314, 1994.

[36]Y.-M. Yuan, S. Wolhauser, M. Mller, J. Klackenberg, M.W. Callmaander, and P. Küpfer. Phylogeny and biogeography of exacum (gentianaceae): a disjunctive distribution in the indian oceaan basin resulting from long distance dispersal and extensive radiation. *Systematic Biology*, 54(1), 2005.