*Syst. Biol.* 50(1):7–17, 2001

# Stochastic Search Strategy for Estimation of Maximum Likelihood Phylogenetic Trees

LAURA A. SALTER[1] AND DENNIS K. PEARL[2,3]

[1]*Department of Mathematics and Statistics, University of New Mexico, Albuquerque, NM 87131, USA; E-mail: salter@stat.unm.edu*

[2]*Department of Statistics, The Ohio State University, Columbus, OH 43210, USA; E-mail: dkp@stat.ohio-state.edu*

*Abstract.*—The maximum likelihood (ML) method of phylogenetic tree construction is not as widely used as other tree construction methods (e.g., parsimony, neighbor-joining) because of the prohibitive amount of time required to find the ML tree when the number of sequences under consideration is large. To overcome this difficulty, we propose a stochastic search strategy for estimation of the ML tree that is based on a simulated annealing algorithm. The algorithm works by moving through tree space by way of a "local rearrangement" strategy so that topologies that improve the likelihood are always accepted, whereas those that decrease the likelihood are accepted with a probability that is related to the proportionate decrease in likelihood. Besides greatly reducing the time required to estimate the ML tree, the stochastic search strategy is less likely to become trapped in local optima than are existing algorithms for ML tree estimation. We demonstrate the success of the modified simulated annealing algorithm by comparing it with two existing algorithms (Swofford's PAUP* and Felsenstein's DNAMLK) for several theoretical and real data examples. [Maximum likelihood; simulated annealing; stochastic probing; stochastic search.]

A great variety of methods for estimating phylogenetic trees from DNA or RNA sequence data are currently available. Among these, the maximum likelihood (ML) method has several advantages, including statistical consistency (Felsenstein, 1981; Hasegawa et al., 1991; Yang, 1994; Chang, 1996; Rogers, 1997), robustness to violations in the assumptions of the underlying evolutionary models (Hasegawa et al., 1991; Yang, 1994), and the possibility of significance tests that use the likelihood framework (Kishino and Hasegawa, 1989; Goldman, 1993). However, use of the ML method in practice has been limited by two difficulties in its implementation. First, existing algorithms are extremely time-consuming and therefore are not easily used when the number of sequences under consideration is large (say, $\geq$35). Second, these algorithms can become trapped in tree topologies that are local maxima when even a moderately large number of DNA sequences are considered (see, for example, Olsen et al., 1994).

The most popular algorithms for estimating the ML phylogenetic tree are those developed by Felsenstein (1993) in his programs DNAML and DNAMLK and by Swofford (1998) in PAUP*. In the case of unrooted trees, Olsen et al. (1994) provided an algorithm (fastDNAML) that is nearly identical to Felsenstein's program DNAML but can reach an estimate more quickly. These algorithms are similar and involve, first, building an initial tree by stepwise addition of sequences to the tree and then moving through tree space deterministically, trying various rearrangements of the current tree. If a rearrangement that improves the likelihood is found, then that tree becomes the current tree. The process continues until no rearrangements of the current tree result in a tree of greater likelihood. Such methods can easily lead to entrapment in local optima. Indeed, simply ordering the sequences differently at the stage of stepwise addition can result in different estimates of the ML tree (Felsenstein, 1993).

Considerable recent attention has been given to stochastic search strategies as a means of estimating the ML tree. The genetic algorithms of Lewis (1998) for nucleotide sequence data and of Matsuda (1996) for amino acid sequence data seem promising. Several Markov Chain Monte Carlo (MCMC) methods (Mau et al., 1996; Yang and Rannala, 1997; Li et al., 2000) have been proposed when an estimate of the posterior distribution of phylogenetic trees under specific prior assumptions is desired. However, when a large number of sequences is considered, MCMC methods give poor estimates of the posterior probability of any individual tree and are not designed to estimate the ML tree. Here

[3]Author for correspondence.

we propose a new stochastic search strategy for estimating the ML tree that is based on a simulated annealing algorithm. After describing the general simulated annealing algorithm, we provide a detailed explanation of our particular implementation of the algorithm. Then we illustrate the application of the algorithm to theoretical and real data examples, respectively, and compare the algorithm with the ML algorithms of Swofford and Felsenstein in both cases. We conclude with a discussion of these results and some directions for future research.

## Modified Simulated Annealing for Estimating the ML Phylogenetic Tree

The method of simulated annealing was developed through the generalization of an algorithm proposed by Metropolis et al. (1953) to simulate the changes in the energy of a substance as it is being cooled. The goal of the cooling process is to obtain a solid that is in its ground state, that is, the state at which the solid has minimum energy. However, the cooling process has the property that if the temperature is lowered too quickly, the resulting solid can become trapped in a metastable state that is not its ground state. Several authors, notably Kirkpatrick et al. (1983) and Cerny (1985), noticed the analogy between the cooling of a substance to its minimum energy state and the minimization of a function by using a stochastic search strategy. In this analogy, the metastable states represent local minima, the ground state represents the global minimum, and the rate of lowering of the temperature corresponds to some parameter that controls the possible solutions examined by the search procedure.

To describe the annealing algorithm in general, we suppose that the goal of the algorithm is the maximization of a function. The algorithm works by moving through the solution space in such a way that solutions that increase the value of the objective function are always accepted whereas those that decrease the value of the objective function are accepted with some probability, the value for which depends on the amount by which the objective function would be decreased. The probability of accepting a solution that lowers the value of the objective function is decreased as the algorithm proceeds according to a sequence of control parameters. This is analogous to the method of slowly decreasing the temperature in the cooling of a substance in the chemical physics setting described above. The idea behind the algorithm is that accepting poorer solutions with a certain probability will help the process avoid becoming trapped in local maxima.

We can formally define the simulated annealing algorithm as follows. Let $f(\cdot)$ be the function to be maximized, let $x_i$ and $x_j$ be elements of the solution space $S$, and let $c_0$ be the initial value of the control parameter. The following steps are repeated until the value of the control parameter is sufficiently small or until the same solution is repeatedly generated in many consecutive iterations.

Step 1. From the current solution, $x_i$, generate a potential solution, $x_j$, according to a specific generation scheme.

Step 2. If $f(x_j) \geq f(x_i)$, then set $x_{i+1} = x_j$. Otherwise, set $x_{i+1} = x_j$ with probability

$$\exp\left\{\frac{-(f(x_i) - f(x_j))}{c_i}\right\}.$$

Step 3. Update the value of the control parameter, $c_i$, and set $i$ to $i + 1$. Go to Step 1.

Because the outcome of an iteration depends only on the outcome of the previous iteration, the simulated annealing algorithm generates a time-inhomogeneous Markov chain, the transition probabilities of which depend on the generation scheme and on the probability of accepting a new solution into the chain. By Markov chain theory, the above algorithm can be shown to converge to a stationary distribution for which the set of optimal solutions has probability one, under certain conditions of both the sequence of control parameters and the generation scheme (Lundy and Mees, 1986; Mitra et al., 1986; Aarts and Korst, 1989; Haario and Saksman, 1991).

We note that the algorithm requires several things. First, a method of generating a candidate solution from any given solution must be determined, and this generation scheme must satisfy the requirement that every state is reachable from every other state in a finite number of applications of the generation procedure. Next, an initial value of the control parameter and the way that the control parameter will be updated at each iteration of the algorithm must be specified. This is often called the *cooling schedule*, in the spirit of the analogy of the cooling of a substance, because this sequence of parameters determines the change in the probability with which the solutions that decrease the value of the objective function are accepted (corresponding to the rate at which the substance is cooled). Finally, the stopping criteria must be determined. This is usually done by specifying either a final value of the control parameter or a bound on the number of consecutive unsuccessful moves attempted.

Although the algorithm can be shown to converge under appropriate conditions, the number of iterations required to find a solution arbitrarily close to the optimal solution can be quite large for some choices of the cooling schedule and stopping rule. Therefore, several cooling schedules that seem to provide near-optimal results in reasonable time have been proposed (Lundy, 1985; Lundy and Mees, 1986; Kirkpatrick et al., 1983; Mitra et al., 1986; Aarts and Van Laarhoven, 1985). Here, we adopt the approach of Lundy (1985) and Lundy and Mees (1986).

Simulated annealing has previously been applied to the phylogenetic tree problem under the parsimony criteria. Barker (1997) implemented a simulated annealing algorithm for estimating parsimonious phylogenetic trees in his program LVB. Several other authors (Lundy, 1985; Dress and Kruger, 1987) have applied simulated annealing to the Steiner tree problem.

A modified version of the simulated annealing algorithm will be used here to estimate the ML phylogenetic tree, so that the function $f$ we are trying to optimize is the log likelihood of the tree. We begin with the problem of finding a rooted tree for which the times for the internal nodes of the tree will be estimated. The length of the branch connecting two nodes is then the difference between the times for the two nodes. We assign a time of 0 to the root node and let the time for each of the external nodes be equal to the total time represented by the tree, which is a parameter to be estimated. Times are then assigned to the internal nodes according to their distance from the root node. The units of time are the expected number of nucleotide substitutions per site. A computer program was written to carry out all of the details of the method proposed here. The program is general enough to include the most popular substitution models in the calculations: the F84 model (Felsenstein, 1993), the HKY85 model (Hasegawa et al., 1985), and all of their associated submodels. The choices we have made for the various components of the annealing algorithm are described below.

### *The Candidate Tree*

To specify the simulated annealing procedure, we must first determine the generation scheme, that is, the method by which a new candidate solution will be generated from an existing solution. The generation scheme proposed here is a stochastic modification of the Nearest Neighbor Interchange (NNI) strategy, used by the PAUP* software (also called the "local rearrangement" strategy in the stochastic search methods used by Kuhner et al. [1995] and Li et al., [2000]). This generation scheme satisfies the requirement that any tree be accessible from any other tree in a finite number of applications of the process (Li et al., 2000). The steps involved in the local rearrangement strategy in this setting for a tree that includes n sequences are as follows (see Fig. 1 for an illustration):

Step 1. From the set of $n - 2$ internal nodes (excluding the root), select one of the internal nodes at random (each equally likely). Call this node the target node.

Step 2. From the set containing the two children nodes of the target node and the sibling of the target node, randomly select two of the three members of the set to become the new children of the target node. The subtrees descending from these children are carried along in the rearrangement process.

Step 3. Generate a new time for the target node.

Note that the time of the target node is constrained to fall between the time of the target's parent node and the time of the closer of the two new children. To assign a time to the target node after local rearrangement, we draw a random number from a beta distribution for which the mean is suggested by a single-step Newton-Raphson (NR1) estimate of the time. The actual value of the mean is

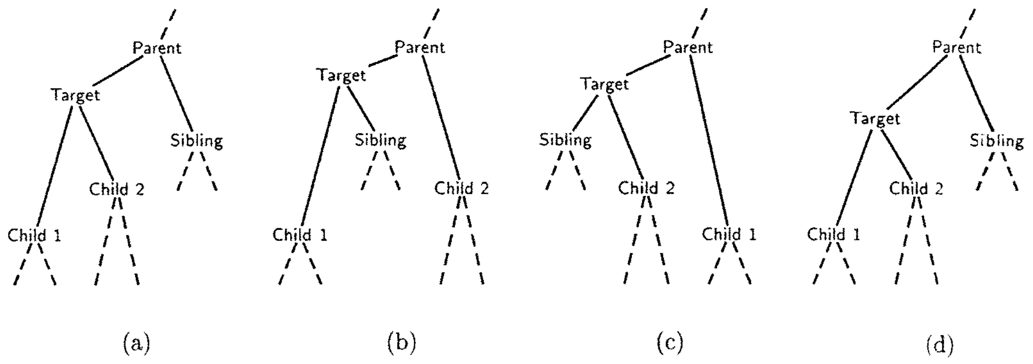

FIGURE 1. Illustration of the local rearrangement strategy. First, a target node is chosen from the set of internal nodes, and the children, sibling, and parent of the target node are identified (a). Two new children of the target node are then selected, resulting in trees (b) or (c), in which the topology has changed, or in tree (d), in which the topology remains the same. The time of the target node is changed regardless of which new tree topology is produced.

set to be two-thirds of the distance from the current time of the target node to the NR1 estimate, if this estimate falls within the allowable interval, or two-thirds of the distance to the endpoint of the interval in the direction of the NR1 estimate otherwise. For computational speed, the NR1 estimate is based only on the subtree descending from the target node. This simplifies the computation enormously, because by the pruning algorithm of Felsenstein (1981), the likelihood of the subtree depends only on the likelihoods of the two children, which have already been calculated. The first and second derivatives of the likelihood of the subtree are thus easily calculable. The variance of the beta distribution used to place the time within the allowable interval is decreased as the algorithm proceeds according to the formula

$$\frac{1}{w + \frac{1}{n}\sqrt{i}},$$

where $i$ is the number of iterations in the annealing algorithm, and $w$ must be $\geq 51$ to ensure that a unimodal beta distribution is obtained. The rationale behind decreasing the variance of the beta distribution is that we rely more and more on the information from the NR1 calculation as the algorithm proceeds. Note that in this step we have generalized the simulated annealing algorithm somewhat, because most instances of the algorithm have generation probabilities that depend only on the current solution. Because our generation probabilities depend also on the iteration number, our stochastic optimization algorithm shares some of the features of the stochastic probing algorithm described by Laud et al. (1992).

For several reasons, we do not spend too much effort on the estimation of the node times and have chosen to estimate only one node time at each iteration. First, the topology of the tree contributes more to the overall log likelihood than do the node times within the tree. In addition, having a tree with node times that have been chosen to maximize the likelihood could result in the rejection of moves that would eventually be beneficial, because the probability of accepting a move depends on the difference in log likelihood between the candidate tree and the proposed tree. Finally, optimization of node times is computationally intensive and would slow the algorithm, making it less applicable to large data sets (Olsen et al., 1994). Despite placing only a limited effort on the estimation of optimal node times, the algorithm generally moves toward the optimal estimates as it proceeds, because of both the decreasing variance of the beta distribution used to generate the new node times and the positive selective pressure placed on "good" node times by the annealing algorithm. The final trees reported by the algorithm will typically have node times that are close to the optimal values, though they will not be exact because of the stochastic nature of the algorithm.

A tree on which to start the local rearrangement procedure must also be specified. Given that convergence of the annealing algorithm occurs independently of the starting point, we choose to start with a tree topology that is randomly generated from the distribution of topologies created by the Yule model with $n$ external nodes (Aldous, 1996). The total time represented by the tree is initially set to 1.0, and initial node times are spaced evenly throughout the tree. That the convergence of the algorithm as implemented here is independent of the starting point will be examined with several examples in the next section.

### *The Cooling Schedule*

The next component of the annealing algorithm that must be specified is the cooling schedule. In this setting, the cooling schedule of Lundy (1985) and Lundy and Mees (1986) is adopted. The updating of the control parameter is accomplished by setting

$$c_{i+1} = \frac{U}{1 + i\beta}$$

where $U$ is an upper bound on the change in the likelihood in one application of the generation scheme, and $\beta$ is the parameter that controls the rate of cooling; its value is $<1$. Values of $\beta$ close to 1 will result in faster cooling but will increase the chance of becoming trapped in local maxima, whereas values of $\beta \ll 1$ will result in slower cooling, which can lead to longer run times.

To set reasonable values of $\beta$ and U, we run the algorithm for an initial "burn-in" period. Once this period has been completed, we set $U$ to the maximum change in the log likelihood observed during the burn-in. The initial value of the control parameter, $c_0$, is set to $U$, and $\beta$ is determined by

$$\beta = \frac{c}{(1-\alpha)n + (\alpha)\frac{-\ln l}{m}},$$

where $n$ is the number of sequences, $m$ is the number of sites, $c$ and $\alpha$ are between 0 and 1, and $\ln l$ is the log likelihood of the UPGMA tree (Li, 1997). We note that the term $\frac{\ln l}{m}$ represents, on average, the log likelihood of each site in the sequence for a tree that will generally have a log likelihood close to that of the ML tree. The motivation for setting $\beta$ in this manner is that the difficulty of the problem is affected by both the number of sequences in the tree and the magnitude of the log likelihood attributed to each site (lower likelihoods imply less phylogenetic information). Because lower values of $\beta$ provide for slower cooling, which is desirable in more difficult problems, $\beta$ is related to the reciprocal of a linear combination of these two factors. The constant $c$ in the numerator allows the user to alter the rate of cooling without changing the particular linear combination selected in the denominator. For most problems, including those considered here, $c = \alpha = 0.5$ seems to work well.

### *The Stopping Rule*

Although many different stopping criteria are possible, the rule used here specifies that the algorithm terminates when reaching the bound placed on the number of iterations made since the proposal of an "uninvestigated" topology. An uninvestigated topology is a topology that either has never been previously proposed or has not been eligible for local rearrangement an adequate number of times. The number of times a topology must be eligible for local rearrangement to be classified as "investigated" is determined by setting a bound on the probability that an internal node would not have been selected as the target node in the local rearrangement. For a tree with $n$ sequences, there are $n-2$ internal nodes eligible for local rearrangement; thus, the probability that an arbitrary internal node is not selected for local rearrangement in $xn$ trials is $(1 - \frac{1}{n-2})^{xn}$. Specifying this probability will therefore determine $x$. The default in our implementation is to set this probability to 0.05. The bound on the number of iterations made since the proposal of an uninvestigated topology is also taken to be a multiple of $n$. The default we have set in this case is *10n* iterations.

To use this stochastic search method for inference of the ML tree, the likelihood must be recalculated after every local rearrangement and compared with the previous likelihood. However, this does not require complete recalculation of the likelihood from the tips of the tree. Because of the pruning algorithm (Felsenstein, 1981), the only nodes for which likelihoods will change after rearrangement are the target and the target's ancestors; the likelihood of the rest of the nodes in the tree will not be affected. Therefore, storing the likelihoods for all the nodes in the tree at each step in the algorithm and recalculating only likelihoods for nodes affected by a local rearrangement presents a considerable economy in terms of computation time.

Additionally, the program has an option that allows for the $k$ best trees encountered by the algorithm to be saved and printed to a file in Newick format, where $k$ can be set by the user. This allows the program to obtain information about competing trees of high likelihood, which is an additional advantage over existing algorithms that typically provide only a single estimate of the ML tree. (Although PAUP* has the option to save the $k$ best trees found, this option is not currently available when the "addseq = random" option is used.) As mentioned previously, estimates of the node times for these $k$ trees will generally be close to optimal, though they will not be exact. Therefore, node times for these trees should be subsequently optimized.

To summarize, our stochastic search algorithm (SSA) for the ML phylogenetic tree reconstruction problem consists of the following steps (where $L(\cdot)$ is the log likelihood function):

Step 1. From tree $\tau_i$, generate candidate tree $\tau^*$, using the local rearrangement strategy.

Step 2. If $L(\tau^*) \geq L(\tau_i)$, set $\tau_i$ to $\tau^*$. Otherwise, set $\tau_i$ to $\tau^*$ with probability

$$\exp\left\{\frac{L(\tau^*) - L(\tau_i)}{c_i}\right\}.$$

Step 3. Set $c_{i+1} = \frac{U}{1+i\beta}$, $\tau_{i+1} = \tau_i$, and $i = i + 1$. If the stopping criteria are satisfied, then return the $k$ best trees found by the algorithm and proceed to Step 4. If the stopping criteria are not satisfied, return to Step 1.

Step 4. Optimize node times for the $k$ best trees found by the algorithm, and determine the ML estimate.

## Theoretical Examples

To test our stochastic optimization method and to compare the method with the two most popular existing ML programs (Swofford's PAUP* and Felsenstein's DNAMLK), we deemed it desirable to create a data set with a known ML. To do this, we specified data when the number of sequences, $n$, is $2^l, l = 2, 3, \ldots, 7$, so that a completely symmetric tree must be the ML tree under the Jukes–Cantor (JC) model (Jukes and Cantor, 1969). For each $n$, the number of sites in the data set was 128. Ten trials for each data set were performed by PAUP*, DNAMLK, and our SSA, with different starting points. For PAUP* and DNAMLK, the use of different starting points meant that different orderings of the sequences were used (i.e., the "jumble" option was used in PHYLIP, and the option "addseq = random" was used in PAUP*), whereas SSA was started from different random trees as described above. For DNAMLK, the global rearrangement option was off. PAUP* was tested with use of both NNI and TBR branch-swapping. SSA was run with two different cooling schedules: $\alpha = c = 0.50$, which is the recommended cooling schedule, and $\alpha = 0.50$ and $c = 0.25$, which will result in longer run times than the recommended schedule, but allows for a more extensive search. SSA was also modified to provide a "stochastic uphill search" by accepting only trees that increased the log likelihood. Examination of the search results obtained by using this modification allowed us to separate the effect of randomly performing local rearrangements from the effect of the probabilistic acceptance of trees with lower log likelihoods. The number of times the ML topology was recovered for each value of $n$ was recorded for each of the methods, as well as the cpu time required for each trial (Table 1). All computations were performed on a Sun Ultra10 running Solaris 7.0 (SPECfp95 = 22.7; see http://www.spec.org for more information on SPEC benchmarks).

The cpu time (averaged over the runs) used by the programs for each value of $n$ is shown in Table 1. We observe that SSA is faster than PAUP* for $n \geq 32$ and faster than DNAMLK for $n \geq 8$. The difference is especially notable for the largest example attempted, $n = 128$, where SSA took an average of ~3 min for the recommended cooling schedule, compared with ~40 min for PAUP* using NNI branch-swapping and ~1.9 hr for DNAMLK. Figure 2 plots the log of the cpu times in seconds against the log of the number of sequences for each of the methods. Felsenstein (1993) states that the time required by DNAMLK is proportional to the cube of the number of sequences, which is supported by the fact that the slope of the line representing the times for DNAMLK

TABLE 1. Summary of the results of all of the examples discussed in the text. All results are averages over the total number of trials.

| Data set | Time (seconds) | | | | | | Number correct[a] | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SSA ($\alpha = 0.5$, $c = 0.5$) | SSA ($\alpha = 0.5$, $c = 0.25$) | Stochastic uphill search | PAUP* (NNI) | PAUP* (TBR) | DNAMLK | SSA ($\alpha = 0.5$, $c = 0.5$) | SSA ($\alpha = 0.5$, $c = 0.25$ | Stochastic uphill search | PAUP* (NNI) | PAUP* (TBR) | DNAMLK |
| Theoretical data | | | | | | | | | | | | |
| $n = 4$ | 0.08 | 0.10 | 0.01 | 0.01 | 0.01 | 0.04 | 10/10 | 10/10 | 1/10 | 10/10 | 10/10 | 10/10 |
| $n = 8$ | 0.24 | 0.26 | 0.02 | 0.10 | 0.13 | 0.66 | 10/10 | 10/10 | 10/10 | 10/10 | 10/10 | 10/10 |
| $n = 16$ | 0.97 | 1.38 | 0.73 | 1.24 | 4.03 | 7.27 | 10/10 | 10/10 | 10/10 | 10/10 | 10/10 | 10/10 |
| $n = 32$ | 4.16 | 7.22 | 2.79 | 15.86 | 84.90 | 71.90 | 10/10 | 10/10 | 10/10 | 10/10 | 10/10 | 10/10 |
| $n = 64$ | 27.72 | 41.86 | 11.24 | 159.58 | 1,295.02 | 742.83 | 10/10 | 10/10 | 4/10 | 10/10 | 10/10 | 6/10 |
| $n = 128$ | 192.24 | 346.72 | 44.59 | 2,394.64 | 24,112.10 | 6,985.96 | 10/10 | 10/10 | 1/10 | 10/10 | 10/10 | 0/10 |
| Real data | | | | | | | | | | | | |
| mtDNA[b] | 110.50 | 156.12 | 27.41 | 87.38 | 433.16 | 235.06 | 10/10 | 10/10 | 8/10 | 10/10 | 10/10 | 10/10 |
| HPV data[c] | 2,233.43 | 4,004.68 | 452.30 | 6,616.40 | 78,745.60 | 11,402.68 | 9/10 | 10/10 | 3/10 | 2/10 | 10/10 | 1/10 |

[a]The number of times the true ML tree is found for the theoretical data sets or the number of times the tree with the greatest known likelihood is found for the real data sets.
[b]$n = 14$; 231 sites.
[c]$n = 30$; 1,379 sites.

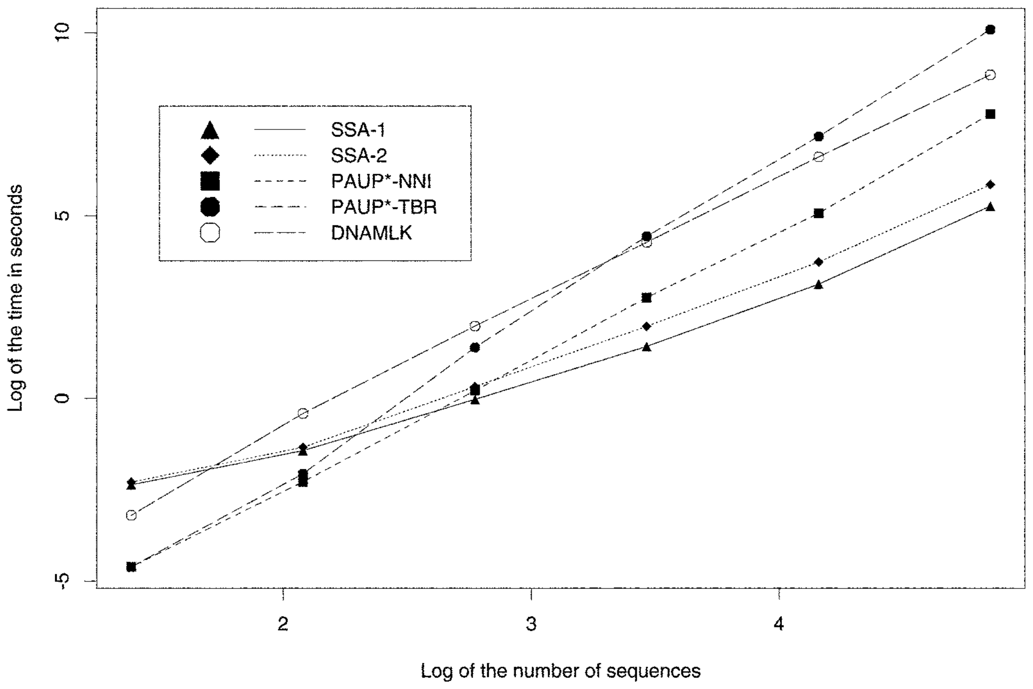

FIGURE 2. Plot of the log of the cpu time in seconds against the log of the number of sequences for SSA with $\alpha = c = 0.50$ (SSA-1), SSA with $\alpha = 0.50$ and $c = 0.25$ (SSA-2), PAUP* with NNI branch-swapping (PAUP*-NNI), PAUP* with TBR branch-swapping (PAUP*-TBR), and DNAMLK from the PHYLIP package (DNAMLK) for the theoretical data. The slopes of the lines, as estimated by least-squares regression, are 2.19 for SSA-1; 2.37 for SSA-2; 3.57 for PAUP*-NNI; 4.30 for PAUP*-TBR; and 3.45 for DNAMLK.

in Figure 2 is ~3. The slope of the line for PAUP* using NNI branch-swapping is also ~3, though the intercept is smaller. For PAUP* using TBR branch-swapping, the slope is ~4. The lines representing the times for SSA under the two cooling schedules that permit probabilistic acceptance of a tree that decreases the log likelihood appear to have slopes of ~2. That is, the time required by SSA is approximately proportional to the square of the number of sequences for the cooling schedule and stopping rule considered here.

We also compared the methods in terms of their ability to recover the ML tree (Table 1). SSA using both cooling schedules and PAUP* using both NNI and TBR branch-swapping recovered the ML tree for all values of $n$, whereas DNAMLK found nonoptimal topologies for $n = 64$ and $n = 128$. This indicates that SSA and PAUP* are better able to avoid entrapment in local maxima than is DNAMLK. Moreover, for SSA, convergence of the algorithm occurs independently of the initial tree.

## REAL EXAMPLES

### *Mitochondrial DNA Sequences*

To test the ability of SSA in a real data example, a data set consisting of mitochondrial DNA sequences with 231 sites for 14 species (see Hayasaka et al., 1988; Salter, 1999) was analyzed by using the F84 model with a transition/transversion ratio of 2.0. Ten trials were performed for SSA, PAUP*, and DNAMLK under the same conditions as described above.

For the cooling schedule with $\alpha = c = 0.50$, SSA performed an average of 3,598 iterations, and took an average of 110.50 sec. In all of the 10 trials, a tree with a log likelihood of −2,677.22, which is presumed to be the ML tree, was listed in the top 10 trees found by the algorithm. Three trees with log likelihoods nearly as high (−2,677.88, −2,677.98, and −2,677.98) were found in the 10 best trees in 9, 10, and 10 of the trials, respectively. Ten other trees appeared in the top 10 trees in more than one trial, with five of these occurring more than five times. In all, 22 unique trees were listed in the 10 best trees over the 10 trials performed.

For the cooling schedule with $\alpha = 0.50$ and $c = 0.25$, SSA performed an average of 4,577 iterations and took an average of 156.12 sec. In all 10 trials, the tree with a log likelihood of −2,677.22, which is presumed to be the ML tree, was listed in the top 10 trees found by the algorithm. The three trees mentioned above with log likelihoods nearly as high were found in the 10 best trees in 7, 10, and 10 of the trials, respectively. Twelve other trees appeared in the top 10 trees on more than one trial, with five of these occurring more than five times. In all, 24 unique trees were listed in the 10 best trees over the 10 trials performed.

Using NNI branch-swapping, PAUP* took an average of 87.38 sec to estimate the ML tree; using TBR branch-swapping, it took an average of 433.16 sec. In both cases, PAUP* always found the ML tree (Fig. 3). DNAMLK took an average of 235.06 sec to estimate the ML tree, and always returned the ML tree (Fig. 3). The three programs always returned the same estimate of the ML tree. PAUP* with NNI branch-swapping is the fastest, which was expected, based on the theoretical data results.

SSA with the recommended cooling schedule takes only about one-fourth of the time required by PAUP* with TBR branch-swapping and about half of the time required by DNAMLK. In addition, SSA gives valuable information about other trees of high likelihood. For example, the tree with a log likelihood of −2,677.88 differs from the ML tree only in reversal of the locations of Bovine and Tarsier.

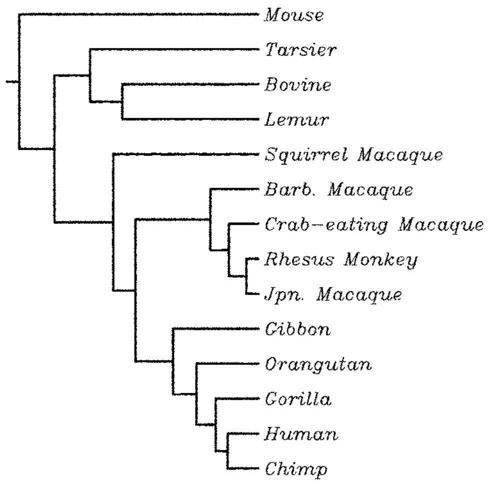

FIGURE 3. Maximum likelihood topology for the 14 sequence mtDNA data set under the F84 model with the transition/transversion ratio at 2.0. The log likelihood of this tree is −2,677.22.

### *Group A9 HPV Sequences*

To test SSA for a larger real data set, we obtained aligned DNA sequences for 30 papillomaviruses (28 human papillomaviruses [HPVs], a rhesus papillomavirus, and a pygmy chimpanzee papillomvirus) from the Los Alamos National Database website (http://hpv-web.lanl.gov) and used a 1,379-nucleotide-long portion of the L1 gene from which all insertion and deletion sites in the sequences had been removed. Similar studies of the evolutionary relationships among papillomaviruses have been conducted by Chan et al. (1992, 1995) and Ong et al. (1997). SSA, PAUP*, and DNAMLK were used to estimate the ML phylogenetic tree under the F84 model with a transition/transversion ratio of 2.0. Ten trials were performed with each algorithm under the conditions previously described.

For the cooling schedule with $\alpha = c = 0.50$, SSA performed an average of 15,261 iterations, which took an average of 2,233.43 sec (~37 min). The highest log likelihood found by the algorithm was a tree with a log likelihood of −27,976.52, which was listed in the top 10 trees found in 9 of the 10 trials. This tree is shown in Figure 4. Two other trees of high likelihood, −27,976.62 and −27,977.67, were often listed in the 10 best trees found by SSA (in 9 and 6 of the 10 trials, respectively). Fourteen other trees were found more than once, of which 5 appeared in 5 or more trials; in all, 28 unique trees were found in the 10 trials.

For the cooling schedule with $\alpha = 0.50$ and $c = 0.25$, SSA performed an average of 27,977 iterations, which took an average of 4,004.68 sec (~1.1 hr). The tree with log likelihood −27,976.52 (Fig. 4) was found in all 10 trials. The two trees of high likelihood mentioned above were found in nine and eight of the trials, respectively. Fifteen other trees were found more than once, of which 6 appeared in 5 or more trials, and a total of 30 unique trees were found in the 10 trials.

Using NNI branch-swapping, PAUP* took an average of 6,616.40 sec (~1.8 hr) and found the ML tree in 2 of the 10 trials. On six of the remaining trials, a tree with a log likelihood nearly as high as that for the ML tree

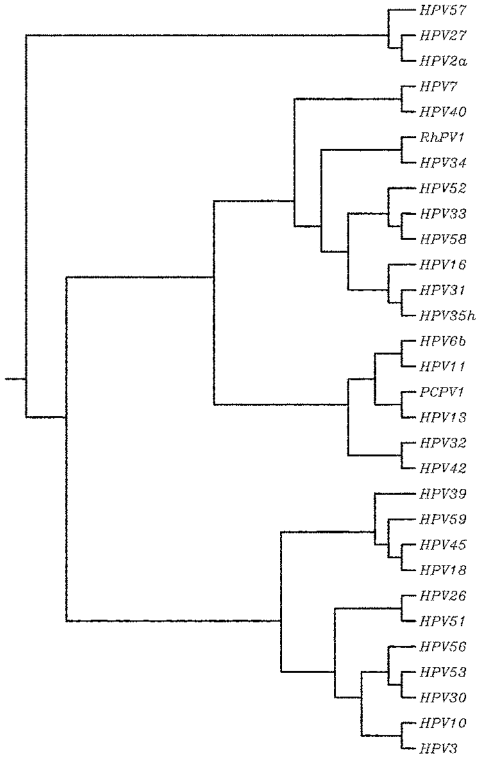

FIGURE 4. Maximum likelihood topology for the HPV data set under the F84 model with the transition/transversion ratio at 2.0. The log likelihood of this tree is −27,976.52.

(−27,977.13) was found. Further study verified that this tree is a local maxima in tree space when NNI moves are used. However, this tree was never listed in the top 10 trees found by SSA. Using TBR branch-swapping, PAUP* took an average of 78,745.60 sec (∼21.9 hr) and always found the ML tree. PAUP* with TBR branch-swapping apparently performs well for this problem but at the expense of increased computing time.

DNAMLK took an average of 11,402.38 sec (∼3.2 hr) to estimate the ML tree, and it returned the ML tree in only 1 of the 10 trials. DNAMLK found the tree with the second largest log likelihood (−27,976.62) on four of the trials, and two trees of reasonably high log likelihood (−27,977.15 and −27,977.24) on two trials each.

SSA is the fastest of the algorithms considered here for this problem and shows good ability to locate the ML tree. In comparison with PAUP* using TBR branch-swapping, which also does a good job of estimating the ML tree, SSA reduced the time required to perform a single estimation from ∼22 hr to ∼1 hr.

## DISCUSSION

Both the theoretical and real data examples in the previous sections demonstrate the usefulness of the SSA proposed here. The method is able to estimate the ML phylogenetic tree much more quickly than PAUP* and DNAMLK for large problems and generally returns an estimate with a likelihood as high as or higher than that returned by PAUP* and DNAMLK. In addition, the algorithm provides some information about alternative trees that also have high likelihood. The advantages of the SSA are expected to be even more substantial as larger data sets are considered. Thus the stochastic search strategy in its present form should be a useful contribution to the collection of algorithms for construction of ML phylogenetic trees.

Comparing the SSA modified to perform an uphill search with SSA under the two cooling schedules that permit probabilistic acceptance of trees with lower values of the log likelihood provides some interesting results. The uphill random search does surprisingly well, even in the case of the $n = 30$ data set, where it outperforms PAUP* with NNI branch-swapping. However, the success of SSA clearly depends on its ability to accept trees that have lower log likelihoods. The fact that the ML tree is found only 3 of 10 times when using the uphill random search demonstrates that failure to permit acceptance of trees with lower log likelihoods will often result in the estimation of trees that are not globally optimal.

The SSA also has the potential to assist in other estimation problems associated with ML trees. For example, SSA as presented here can be easily modified to handle the case of unrooted trees. The only necessary change is that during the application of the local rearrangement strategy to generate new candidate trees, branch lengths, rather than node times, must be generated. Further, the SSA provides a framework within which simultaneous estimation of the tree and of the parameters in the substitution models, such as the transition/transversion parameter and the nucleotide frequency parameters, would be possible. Adapting the annealing algorithm to handle models that allow for site-to-site rate variation or models that incorporate a process of insertion and deletion would also be interesting.

## ACKNOWLEDGMENTS

We thank Joseph Felsenstein for suggesting the 14-sequence mtDNA example for comparative study. We also thank Dave Swofford for providing a beta version of PAUP*.

## REFERENCES

AARTS, E., AND J. KORST. 1989. Simulated annealing and Boltzman machines. Wiley and Sons, New York.

AARTS, E., AND P. VAN LAARHOVEN. 1985. A new polynomial time cooling schedule. Pages 206–208 *in* Proc. IEEE Int. Conf. on Computer-Aided Design, Santa Clara, California, IEEE, New York.

ALDOUS, D. 1996. Probability distributions on cladograms. Pages 1–18 *in* Random Discrete Structures (D. Aldous and R. Permantle, eds.) Springer, New York.

BARKER, D. 1997. LVB 1.0: Reconstructing evolution with parsimony and simulated annealing. Univ. Edinburgh, Scotland.

CERNY, V. 1985. Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. J. Optimization Theory Appl. 45:41–51.

CHAN, S., H. BERNARD, S. ONG, S. CHAN, B. HOFMANN, AND H. DELIUS. 1992. Phylogenetic analysis of 48 papillomavirus types and 28 subtypes and variants: A showcase for the molecular evolution of DNA viruses. J. Virol. 66:5714–5725.

CHAN, S., H. DELIUS, A. HALPERN, AND H. BERNARD. 1995. Analysis of genomic sequences of 95 papillomvirus types: Uniting typing, phylogeny, and taxonomy. J. Virol. 69:3074–3083.

CHANG, J. 1996. Full reconstruction of Markov models on evolutionary trees: Identifiability and consistency. Math. Biosci. 137:51–73.

DRESS, A., AND M. KRUGER. 1987. Parsimonious phylogenetic trees in metric spaces and simulated annealing. Adv. Appl. Math. 8:8–37.

FELSENSTEIN, J. 1981. Evolutionary trees from DNA sequences: A maximum likelihood approach. J. Mol. Evol. 17:368–376.

FELSENSTEIN, J. 1993. PHYLIP (Phylogenetic inference package), version 3.5p. Univ. Washington, Seattle.

GOLDMAN, N. 1993. Statistical tests of models of DNA substitution. J. Mol. Evol. 36:182–198.

HAARIO, H., AND E. SAKSMAN. 1991. Simulated annealing process in general state space. Adv. Appl. Prob. 23:866–893.

HASEGAWA, M., H. KISHINO, AND N. SAITOU. 1991. On the maximum likelihood method in molecular phylogenetics. J. Mol. Evol. 32:443–445.

HASEGAWA, M., H. KISHINO, AND T. YANO. 1985. Dating of the human–ape splitting by a molecular clock of mitochondrial DNA. J. Mol. Evol. 21:160–174.

HAYASAKA, K., T. GOJOBORI, AND S. HORAI. 1988. Molecular phylogeny and evolution of primate mitochondrial DNA. Mol. Biol. Evol. 5:626–644.

JUKES, T. H., AND C. R. CANTOR. 1969. Evolution of protein molecules. Pages 21–132 *in* Mammalian protein metabolism (H. N. Munro, ed.). Academic Press, New York.

KIRKPATRICK, S., C. D. GELATT, JR., AND M. P. VECCHI. 1983. Optimization by simulated annealing. Science 220:671–680.

KISHINO, H., AND M. HASEGAWA. 1989. Evaluation of the maximum likelihood estimate of the evolutionary tree topologies from DNA sequence data, and the branching order in Hominoidea. J. Mol. Evol. 29:170–179.

KUHNER, M. K., J. YAMATO, AND J. FELSENSTEIN. 1995. Estimating effective population size and mutation rate for sequence data using Metropolis–Hastings sampling. Genetics 140:1421–1430.

LAUD, P. W., L. M. BERLINER, AND P. K. GOEL. 1992. A stochastic probing algorithm for global optimization. J. Global Optimization 2:209–224.

LEWIS, P. 1998. A genetic algorithm for maximum-likelihood phylogeny inference using nucleotide sequence data. Mol. Biol. Evol. 15:277–283.

LI, S., D. PEARL, AND H. DOSS. 2000. Phylogenetic tree construction using Markov Chain Monte Carlo. J. Am. Stat. Assoc. 95:493–508.

LI, W.-H. 1997. Molecular evolution. Sinauer Associates, Sunderland, Masschusetts.

LUNDY, M. 1985. Applications of the annealing algorithm to combinatorial problems in statistics. Biometrika 72:191–198.

LUNDY, M., AND A. MEES. 1986. Convergence of an annealing algorithm. Math. Program. 34:111–124.

MATSUDA, H. 1996. Protein phylogenetic inference using maximum likelihood with a genetic algorithm. Pacific Symposium on Biocomputing (L. Hunter, T. E. Klein, editors) Singapore, World Scientific.

MAU, B., M. A. NEWTON, AND B. LARGET. 1996. Bayesian phylogenetic inference via Markov Chain Monte Carlo methods. Tech. Rep. No. 961. Dept. of Statistics, Univ. Wisconsin-Madison.

METROPOLIS, N., A. ROSENBLUTH, M. ROSENBLUTH, A. TELLER, AND E. TELLER. 1953. Equation of state calculations by fast computing machines. J. Chem. Phys. 21:1087–1092.

MITRA, D., F. ROMEO, AND A. SANGIOVANNI-VINCENTELLI. 1986. Convergence and finite-time behavior of simulated annealing. Adv. Appl. Prob. 18:747–771.

OLSEN, G., H. MATSUDA, R. HAGSTROM, AND R. OVERBEEK. 1994. FastDNAml: A tool for construction of phylogenetic trees of DNA sequences using maximum likelihood. Comput. Appl. Biosci. 10:41–48.

ONG, C., S. NEE, A. RAMBAUT, H. BERNARD, AND P. HARVEY. 1997. Elucidating the population histories and transmission dynamics of papillomaviruses using phylogenetic trees. J. Mol. Evol. 44:199–206.

ROGERS, J. 1997. On the consistency of maximum likelihood estimation of phylogenetic trees from nucleotide sequences. Syst. Biol. 46:354–357.

SALTER, L. 1999. Simulation-based estimation of phylogenetic trees. Ph.D. Dissertation, The Ohio State Univ., Columbus.

SWOFFORD, D. L. 1998. PAUP*. Phylogenetic analysis using parsimony (* and other methods), version 4. Sinauer Associates, Sunderland, Massachusetts.

YANG, Z. 1994. Statistical properties of the maximum likelihood method of phylogenetic estimation and comparison with distance matrix methods. Syst. Biol. 43:329–342.

YANG, Z., AND B. RANNALA. 1997. Bayesian phylogenetic inference using DNA sequences: A Markov Chain Monte Carlo method. Mol. Biol. Evol. 14:717–724.

*Received 4 February 1999; accepted 24 June 1999*
*Associate Editor: M. Sanderson*