Representation in stochastic search for phylogenetictree reconstruction

ARTICLE IN PRESS

Available online at www.sciencedirect.com
SCIENCE DIRECT®

Journal of Biomedical Informatics xxx (2006) xxx–xxx

Journal of Biomedical Informatics

www.elsevier.com/locate/yjbin

Representation in stochastic search for phylogenetic tree reconstruction

Griffin Weber [a,b,*], Lucila Ohno-Machado [a], Stuart Shieber [b]

[a] Decision Systems Group, Brigham and Women's Hospital, Division of Health Sciences and Technology, Harvard and MIT, USA
[b] Division of Engineering and Applied Sciences, Harvard University, USA

Received 28 May 2005

Abstract

Phylogenetic tree reconstruction is a process in which the ancestral relationships among a group of organisms are inferred from their DNA sequences. For all but trivial sized data sets, finding the optimal tree is computationally intractable. Many heuristic algorithms exist, but the branch-swapping algorithm used in the software package PAUP* is the most popular. This method performs a stochastic search over the space of trees, using a branch-swapping operation to construct neighboring trees in the search space. This study introduces a new stochastic search algorithm that operates over an alternative representation of trees, namely as permutations of taxa giving the order in which they are processed during stepwise addition. Experiments on several data sets suggest that this algorithm for generating an initial tree, when followed by branch-swapping, can produce better trees for a given total amount of time.

Keywords: Phylogenetic tree reconstruction; Parsimony; Hill-climbing; Stepwise addition; Branch-swapping; Tree bisect reconnect; Three-way labels

1. Introduction

A phylogenetic tree is a graph representing the ancestral relationships among a group of organisms. Phylogenetic tree reconstruction is an approximation to the true phylogenetic tree for a set of organisms based on contemporaneous evidence such as corresponding DNA sequences. The problem is thought of in computational terms as a combinatorial optimization problem. The combinatorial aspect involves a search through the space of all possible such trees; the optimization is defined by an objective function that approximates a notion of the closeness of each potential tree to the correct tree. Although more realistic objective functions such as *maximum likelihood* have been defined and explored, the *maximum parsimony* objective function [1,2]—the summed Hamming distances between edge-connected taxa—is widely used for its computational simplicity. Phylogenetic reconstruction using maximum parsimony involves finding the tree with the lowest score. The advantage of maximum parsimony over maximum likelihood is that it is often computationally simpler. Even so, for n taxa, there are $(2n-5)!!$ possible unrooted trees [1]. Therefore, with any nontrivial number of taxa, the problem of finding the optimal tree is far too complex to solve exactly in a realistic amount of time.

A wide variety of methods for reconstructing phylogenetic trees have been proposed, from simple clustering methods to algorithms that form complex probabilistic models of evolution [1,3–6]. Chief among these is a series of *stochastic search* procedures such as those implemented in the widely used PAUP* system [7]. In general, a stochastic search algorithm for solving a given problem requires specifying a *representation* of candidate solutions and a *search regime*. Previous stochastic search procedures have differed primarily in the search regime used. In this paper, by contrast, we show that by varying the representation of the search space, rather than the search regime, we can achieve improved performance over algorithms using the standard tree representation. In so doing, we introduce a

* Corresponding author. Fax: +1 617 496 1066.
E-mail address: weber@fas.harvard.edu (G. Weber).

1532-0464/$ - see front matter © 2005 Elsevier Inc. All rights reserved.
doi:10.1016/j.jbi.2005.11.001

new heuristic algorithm for maximum parsimony phylogenetic tree reconstruction that outperforms the standard branch-swapping algorithms used by most software programs [7].

The paper begins with an overview of stochastic search and its application to phylogenetic tree reconstruction in Section 2, and describes the branch-swapping algorithm in stochastic search terms and presents the new representation. Section 3 lists the data sets and experiments used to compare the new algorithm to the current standards. Section 4 illustrates the advantages of the new algorithm for these data sets. The actual implementation of these algorithms can have a dramatic effect on performance. Section 5 addresses this issue by describing the optimization techniques that the software written for this study uses. Section 6 summarizes the results and suggests future directions for this research.

2. Stochastic search for phylogenetic tree reconstruction

A stochastic search procedure is a method for solving a combinatorial optimization problem that proceeds by taking a random walk in a space of *representations* of solutions to the problem. The possible steps in the walk are defined by one or more *operators* by which an existing candidate solution is perturbed to form a new solution; the possible steps form a neighborhood structure over the representation space, and hence over the solution space.

A search procedure uses a particular regime for carrying out this random walk. *Hill climbing*, for instance, works as follows: an initial candidate solution is generated, the current solution. Then iteratively, the current solution is perturbed by application of an operator to form a new solution. If the score of the new solution is lower than that of the current solution, the new solution becomes the candidate solution. We take lower scores to be better consistent with the maximum parsimony objective function. After all iterations are completed, the current solution is returned.

The *simulated annealing* search regime also keeps the perturbed candidate if it has a lower score; however, if it has a higher score, it keeps the perturbed candidate with a monotonically decreasing probability determined by an ''annealing schedule'' [8–10]. *Parallel descent* maintains a set of candidate solutions, selecting one to perturb proportionately to its score and keeping it if its score is higher than the current worst candidate, replacing it in the candidate set. And so forth. A wide variety of such search procedures can be defined. Importantly, all operate uniformly over a given space of represented solutions and perturbation operators.

To carry out a stochastic search for a combinatorial optimization problem, then, we require both a representation of the problem (including the solution space and operators) and a search regime. To date, much of the experimentation with alternative stochastic search methods for phylogenetic tree reconstruction has addressed the issue of search regime, with relatively little attention paid to representation. In particular, previous work has used a direct representation of phylogenetic trees as the representation method. In this work, we show that search over an alternative representation based on a greedy decoder, when used together with the traditional representation, can outperform search using either of the two representations taken singly. Before presenting the experimental results, however, we describe the two representations and their use in various search regimes.

2.1. The direct representation

We take the paradigm of stochastic search for phylogenetic tree reconstruction to be the software package PAUP*. For the purpose of defining perturbation operators, PAUP* uses a representation of trees based directly on their tree structure, which we call the *direct representation*. A tree is represented directly by its topology, the nodes and edges connecting them. (The representation is so obvious that it may be surprising that alternatives exist, but one will be described shortly.)

Generation of an initial tree representation can be done simply by *stepwise addition*. An algorithm to do so begins with three taxa joined in a trivial tree with three leaves representing the taxa and one internal node representing the common ancestor. Each additional taxon is then added to the tree by (1) removing an edge, (2) reconnecting that edge's vertices to a new internal node, and (3) connecting the new taxon to the internal node [7]. The RANDOM variant of stepwise addition selects the initial three taxa, the edge to remove, and the new taxon to add at each step randomly. Because no information about the sequences is used to construct the tree, its parsimony score is usually very poor. Alternatively, the taxa can be added in a greedy fashion. The GREEDY variant works on a given permutation of the taxa by forming a tree from the first three taxa, adding each successive taxon at that edge that optimizes the score of the tree built so far. The resulting tree depends deterministically on the order in which the sequences are added to the tree, i.e., on a permutation of the taxa.

Perturbation of a tree represented under the direct representation can be accomplished in various ways. If the perturbed version of the tree is constructed as a separate random tree independent of the tree being perturbed, the hill-climbing search regime degenerates to a particularly simple (and poor) stochastic search method of randomly generating trees and selecting the lowest scoring one; this regime is traditionally referred to as *random generate and test* (RGT). This method is unlikely to find the optimal tree since the search space is very large. Nonetheless, it can serve as a simple benchmark method.

A perturbation method ought ideally to attend to and preserve much of the structure of the tree being perturbed. The standard perturbation method for the direct representation, used by PAUP*, is Tree Bisection and Reconnection (TBR) [7,11]. Given a tree T, TBR involves first

removing an edge to form two subtrees, then creating a new edge that connects the two subtrees in a different arrangement.

TBR under a hill-climbing regime is the basic stochastic search method used by PAUP*. A disadvantage of the hill-climbing regime is that it finds local, rather than global, optima. The success of random descent algorithms often depends on the shape of the "terrain." If there is a smooth path leading to the global optimum, then a good solution is much easier to find than if there are many local optima that form a bumpy surface for the algorithm. The branch-swapping algorithm that PAUP* uses is known to find local minima (i.e., getting "stuck" on "islands" of poor solutions) [12]. However, the best known trees for some data sets have been formed using PAUP* and branch-swapping [13].

PAUP* allows other search regime variants to be used, still with the direct representation and TBR as the perturbation operator [7]. Rather than keeping just the single best tree after each iteration, PAUP* can keep multiple trees. It attempts to improve the score of each tree relying on the fact that if some of the trees fall into poor local minima, then there will be other trees that can continue with better scores. There are different ways of choosing the trees to keep. One method is to start with a certain number of random trees, and improve each tree in parallel. Another method is to delete the worst trees after each full iteration and replace them with duplicates of the trees that are performing best. Note that because the algorithm swaps branches randomly, duplicates of the same tree will likely diverge after a few iterations. The total number of trees retained after each step can be a fixed number, or it can grow or shrink according to pre-defined criteria. Each of these variant search regimes displays slightly different performance on particular problem instances. All share reliance on the direct representation of phylogenetic trees and TBR as the perturbation operator.

PAUP* can also select the permutation used to generate the initial tree in one of four user-specified ways. The *as-is* method uses the order in which the sequences are listed in the input data file. The *simple* method starts with a reference sequence, then at each step chooses the sequence with the smallest Hamming distance to the reference sequence. The *closest* method tries each available sequence, calculates the score for the tree, and chooses the sequence that produces the lowest score. Because it tries every sequence in every possible position, this is by far the most computationally demanding method. It also is completely deterministic given only the sequences, unlike the as-is method that depends on the order of the taxa in the input file or the simple method that depends on the reference sequence. The final method, PAUP*'s default and the one used here, is *random*, which adds the sequences to the growing tree in a randomly selected order. An alternative representation of phylogenetic trees is described below, and provides the basis for a systematic method for determining a good permutation for constructing the initial tree for TBR, one providing substantial improvement even taking into consideration the time costs.

2.2. The greedy decoder representation

Research related to stochastic search algorithms has shown that changing the problem representation can dramatically affect the terrain of the problem and hence the quality of solutions found [14]. This observation can be applied to the phylogenetic tree reconstruction problem by designing alternative representations and corresponding perturbation operators. In this paper, we explore a representation based on a *greedy decoder*, a parameterization of the GREEDY variant of stepwise addition described above. Since any permutation of the taxa determines a tree, by execution of GREEDY, the *permutation itself* can be taken to be a representation of that tree. The score of a permutation is calculated by greedily decoding it into the corresponding tree and calculating the score of that tree. A natural perturbation operator for this representation is the swapping of two (or more) taxa in the permutation.

The potential for advantage of the greedy decoder representation for phylogenetic trees can be easily shown. To get a sense for the difference between the representations, we can select trees randomly under the two representations. The RANDOM variant of stepwise addition constructs random trees under the direct representation. Random trees under the greedy decoder representation can be generated by decoding randomly selected permutations of the taxa. Fig. 1 shows histograms of the scores of trees selected under the two representations, RANDOM and GREEDY, for the rbcL500 data set described below. The mean of the direct trees are so many standard deviations worse than the mean greedy decoder tree that it would be nearly impossible for a search regime over the direct trees to find a better tree in a reasonable amount of time.

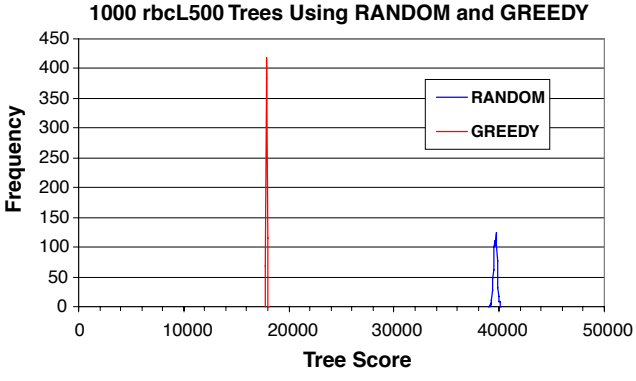

Fig. 1. A histogram showing the frequency of different scores when applying the RANDOM and GREEDY algorithms to the rbcL500 data set 1000 times. Notice the widths of the curves compared to the distance between their means.

3. Performance evaluation methodology

To test empirically the performance of stochastic search over the two representations, experiments were performed using four different stochastic search algorithms to construct phylogenetic trees over a variety of data sets.

3.1. Algorithms

The four algorithms tested varied both in search regime (RGT and hill climbing) and in representation (direct and greedy decoder), resulting in four algorithms.

The RANDOM algorithm performed random-generate-and-test search over the direct tree representation, that is, random trees were generated and the best one saved. The GREEDY algorithm used random-generate-and-test over the greedy decoder representation, that is, random permutations were generated and the one whose decoding was best was saved. These two algorithms served as controls. They are known to produce poor trees; however, they are the two most common starting points for more complex algorithms. The third technique, TBR, was the tree bisection-reconnection hill-climbing algorithm used in PAUP*, with the initial tree formed using a tree generated by the greedy variant of the stepwise addition algorithm.

The final technique, greedy hill climbing (GHC), used random descent over the greedy decoder representation. GHC defines a neighbor in the following way: two distinct taxa are chosen and their order is swapped. Then, another two distinct taxa are swapped, so that the order of up to four taxa can change with one iteration of the algorithm. Although we do not present them here, experiments on the effects of swapping different numbers of pairs of taxa were tested, and swapping two pairs appeared to be a satisfactory choice. Swapping only one pair at a time produced similar quality trees, but required more iterations. Swapping too many pairs effectively eliminated any neighborhood structure, producing results similar to the GREEDY algorithm.

In addition to the four basic algorithms, hybrid methods were also tested, where GHC was run for a certain fraction of the total allotted time, and then TBR was run for the remaining time to further improve the tree. This paper will refer to these hybrid methods as GX:TY, where X and Y represent the relative amounts of time given to each algorithm. For example, G3:T5 runs GHC for the first 3/8 of the total time, and TBR for the remaining 5/8.

Note that the hybrid algorithms must run GHC before TBR. This is because GHC only considers trees that can be formed using stepwise addition. For n taxa, there are $n!$ possible orderings, and therefore at most $n!$ different stepwise addition trees. In contrast, TBR can produce any of the $(2n-5)!!$ possible unrooted trees. As a result, the majority of trees found using TBR cannot be represented as an ordering of sequences, and therefore the output of TBR is not necessarily a valid input to the GHC algorithm.

Table 1
The four data sets used in this study

Data set	Taxa	Length
rbcL500	500	759
Prokaryote	218	1623
Primate	12	898
Simulated	500	25

Shown are the number of taxa in each data set and the length of the sequences.

Although various software packages like PAUP* include the RANDOM, GREEDY, and TBR algorithms, there are no existing versions of GHC or the hybrid methods. Therefore, this study used new software that can apply all of these algorithms to a data set. Having one program that includes all tree reconstruction methods also ensures that time-controlled experiments are not biased by implementation tricks or code optimizations present in one program but not another.

3.2. Data sets

It is well known that there are "good" data sets, where the globally optimal tree can easily be found, and "bad" data sets, with many poor local minima [7]. Unfortunately, with a new algorithm, it is difficult to determine beforehand how a particular data set will perform. Therefore, this study uses a range of sizes and sources for the selected data sets. The first is a 500 taxa database of seed–plant sequences named rbcL500, which is frequently used in phylogenetic tree research [2,5]. The second has 218 taxa (approximately half the size of rbcL500) and consists of sequences obtained from small subunit ribosomal RNA from various prokaryotes. The third is a small 12 taxa primate data set that comes as a sample file with PAUP* and has a known global minimum score. A fourth data set, like rbcL500, has 500 taxa but consists of sequences that were artificially generated by randomly modifying an initial ancestor sequence to simulate the process of evolution (Simulated data set). Speed performance experiments also used several smaller randomly generated synthetic data sets. The three real data sets were modified slightly so that each sequence within a given data set contained the same number of bases, and constant or uninformative bases were removed. Table 1 lists the four primary data sets, the number of taxa, and the actual length of the sequences.

4. Results of experiments

4.1. Controls

Tables 2 and 3 show the results for the RANDOM and GREEDY algorithms after 1000 iterations. For the primate data set, both algorithms found the globally optimal score, which was verified using PAUP*'s branch-and-bound exact search method. However, as noted above,

Table 2
The mean, standard deviation, minimum, and maximum scoring trees for each of the data sets using the RANDOM algorithm

Data set	Mean	SD	Min	Max
rbcL500	39615.13	163.95	39067	40035
Prokaryote	55193.44	289.98	54219	56162
Primate	1566.16	65.53	1327	1688
Simulated	3834.32	22.33	3754	3904

Table 3
The mean, standard deviation, minimum, and maximum scoring trees for each of the data sets using the GREEDY algorithm

Data set	Mean	SD	Min	Max
rbcL500	17881.04	39.32	17769	18039
Prokaryote	35062.80	55.22	34882	35273
Primate	1154.11	0.87	1154	1162
Simulated	951.94	10.63	923	991

GREEDY trees well outperformed RANDOM trees for the larger data sets.

It is important to note, however, that the GREEDY algorithm only tests a very small subset of all possible trees. With n taxa, there are $n!$ possible orderings for stepwise-addition. This is in contrast to the $(2n-5)!!$ distinct unrooted trees that are actually possible with n taxa. This small subset of trees clearly contains trees with some of the best scores. However, there is no guarantee that the optimal tree is within this subset. Therefore, given enough time, the RANDOM algorithm will find the optimal tree, while the GREEDY algorithm might not. Similarly, GHC, because it simply walks through the set of greedy-decoder representable trees, might not find the globally optimal tree. TBR can consider a larger set of trees than GHC, but it is not guaranteed to see all possible trees when greedy stepwise addition is used for the initial trees. Branch-swapping starting from random trees, however, would be more likely to search all trees.

4.2. GHC, TBR, and hybrid algorithms

Because one iteration of TBR involves simply rearranging an existing tree, while one iteration of GHC constructs an entire tree from scratch, many more TBR steps can be performed in a given amount of time. Therefore, controlling the total run time rather than the number of iterations is a more accurate way of comparing TBR and GHC. This is also a reasonable way to measure the algorithms because, in practice, the amount of time the user is willing to wait will determine the limit on how good a tree these algorithms will produce.

The next set of experiments allowed the rbcL500, Prokaryote, Primate, and Simulated data sets to run for a total of 1800, 3600, 8, and 64 s, respectively. These values were based on the number of taxa and the sequence lengths in each data set. Hybrid experiments divide the total time into a GHC component and a TBR component. The experiments compared nine variations: GHC only, G7:T1, G6:T2, G5:T3, G4:T4, G3:T5, G2:T6, G1:T7, and TBR only. For each variation, time versus best-tree-score curves was generated, and the overall best tree after t seconds was recorded.

The actual experiments began with 100 iterations of GREEDY to obtain a starting tree T_0. (The GREEDY algorithm is fast enough that this step requires a negligible amount of time, and it is not included in the total run time.) Then GHC was run for t seconds. The best tree after $t/8$ s was T_1, the best after $2t/8$ s T_2, and so forth until the final tree after t seconds, T_8. Next, TBR was run for t seconds starting with T_0, for $7t/8$ s starting with T_1, $6t/8$ s starting with T_2, and so on, running for only $t/8$ s starting with T_7. In the end, nine trees were obtained, with each one being generated using a different fraction of GHC and TBR over a total of t seconds.

The rbcL500 and Prokaryote experiments were repeated four times using four different starting trees T_0. The Primate and Simulated experiments were repeated a total of 16 times. Fig. 2 illustrates the average tree score for each of the nine algorithms when applied to the rbcL500 data set. The results for the GHC-only algorithm are represented in blue, the TBR-only algorithm in red, and the best hybrid algorithm, which in this case was G5:T3, in green. The results for other hybrid algorithms are represented in gray. Fig. 3 depicts the curves with error bars representing 95% confidence intervals. Between TBR and GHC, there is no clear winner. Which one produces a better tree depends on how much time is allocated. However, Fig. 4 shows that every hybrid algorithm produces better final trees than either TBR or GHC alone.

The results were similar for the Prokaryote and Simulated data sets. All of the hybrid algorithms produce better trees than either TBR or GHC alone. For the Prokaryote data set, the best hybrid was G7:T1, and for the Simulated data set the best hybrid was G5:T3. The Primate data set is so small that most GREEDY trees find the globally optimal solution. Since a GREEDY tree is used as the initial

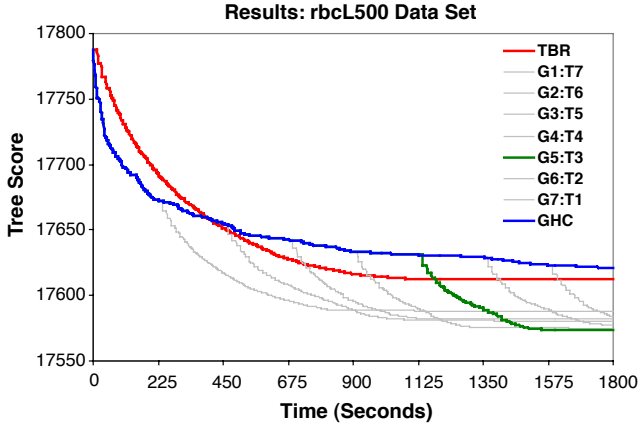

Fig. 2. The scores of the best trees obtained, for the rbcL500 data set, after a given number of seconds for each of nine combinations of tree bisection-resection (TBR) and greedy hill-climbing on stepwise-addition (GHC). The green curve is the hybrid algorithm that produced the lowest-scoring final tree.

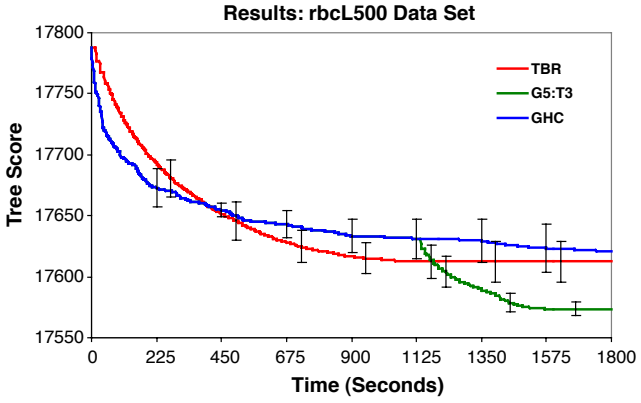

Fig. 3. The scores of the best trees obtained, for the rbcL500 data set, using TBR, GHC, and the best hybrid algorithm, G5:T3. The error bars represent 95% confidence intervals of four runs.

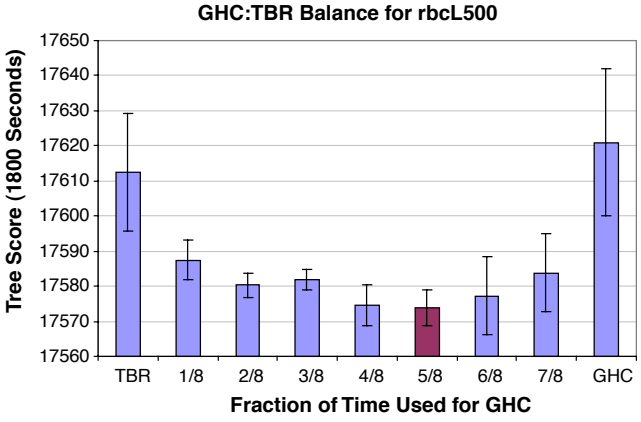

Fig. 4. The scores of the best trees obtained, for the rbcL500 data set, after 1800 s using nine different algorithms. The best algorithm, G5:T3 is highlighted.

tree for the nine variations of GHC and TBR, they also find the globally optimal tree even before the first iteration.

4.3. Uniform time experiments

The experiments in the previous section gave each data set a different amount of time. This was to adjust for the rates in which the algorithms run on the different data sets. The next set of experiments compare the algorithms using 30-min runs for all four data sets. Table 4 lists the number of trees that the TBR and GHC algorithms examined. In each case, TBR considers far more trees; however, the previous results showed that using more trees does not necessarily result in better trees.

Table 4
The number of trees examined in 30 min by the TBR and GHC algorithms for each of the four data sets

Data set	TBR	GHC
rbcL500	1.1×10^9	5438
Prokaryote	4.1×10^8	6211
Primate	8.4×10^7	787725
Simulated	1.1×10^{10}	64120

Table 5
The mean tree score after 30 min using TBR, GHC, G4:T4, or the best hybrid algorithm for each of the four data sets

Data set	TBR	GHC	G4:T4	Best
rbcL500	17613	17621	17574.5	17573.75
Prokaryote	34686	34742	34658.25	34655
Primate	1154	1154	1154	1154
Simulated	900.5	892.25	892	892

Table 5 shows the scores of TBR and GHC after 30 min. With both the rbcL500 and the Prokaryote data sets, TBR was ahead of GHC after 30 min. GHC produced better trees with the Simulated data set. The Simulated data set also allowed 10 times as many iterations of GHC as those allowed in the rbcL500 and Prokaryote data sets for the same amount of time. The best hybrid algorithm for each data set was better than either TBR or GHC alone. However, the optimal ratio of GHC to TBR does not have to be known to produce good trees. Simply splitting the time in half (the G4:T4 algorithm) produced better trees than TBR or GHC alone.

Although the best hybrid algorithm ran in 30 min, it took several times longer to determine which G:T ratio was the best. That same amount of time could have been used to extend any one of the hybrid algorithms to many more iterations, and likely produce an even better tree. It would therefore be desirable to be able to predict the best ratio without having to try all possible algorithms. Notice that in Fig. 3 the scores of GHC gradually drop for the entire duration of the experiment. In contrast, the scores from TBR appear to drop rapidly then abruptly level off. Furthermore, the slope of the TBR component of the best hybrid algorithm mirrors the slope of the TBR only algorithm. In all data sets tested, the best hybrid algorithms are the ones where GHC is run for as long as possible, while leaving enough time at the end for TBR to reach its leveling off stage. One hypothesis is that the optimal hybrid strategy can be determined by first running TBR alone for a short amount of time and observing its rate of descent. This should provide an indication of how much time is needed for TBR at the end of a hybrid algorithm. The experiment would then be restarted using a hybrid algorithm that takes this into account. Although some time would be lost with the initial TBR, having a good prediction for the best hybrid algorithm might help in the end. Evidently, further analysis with additional data sets would have to be performed to test this hypothesis.

5. Implementation considerations

All algorithms described in this paper require comparing the scores of a large number of trees. Because the performance of GHC and TBR depend on the number of iterations possible in a given amount of time, an efficient tree scoring method is essential. Given the sequences at each

node in a tree, the score of the tree can be calculated by following each edge and counting the number of positions (bases) in which adjacent sequences differ. For n taxa and sequences containing m bases, this is $O(nm)$. However, only the leaves of the tree are actually known. The leaves represent existing, or observed species. All $n-2$ internal nodes represent ancestral species whose sequences are unknown. However, for maximum parsimony, the internal sequences are assumed to be those that minimize the overall score of the tree. Luckily, there is a dynamic programming algorithm that can determine the optimal internal sequences and score the tree in $O(rnm)$, where r is the size of the alphabet used in the sequences. For DNA, which consists of the bases {A, C, G, T}, r is 4. Therefore, the time needed to score a tree, even when the internal sequences are unknown, is still simply $O(nm)$.

Consider the complexity of one iteration of the GREEDY algorithm. To perform stepwise-addition, n sequences must be added, one at a time, to a growing tree. Each sequence is placed in $O(n)$ positions and tested before its final position is chosen. Therefore, the scores of $O(n^2)$ trees must be calculated. Thus, the total complexity is $O(n^3m)$. For large data sets, such as the $n=500$ taxa in rbcL500, this algorithm is far too costly. Ganapathy recently described a new technique for scoring a tree, called Three-Way Labels, which works in $O(m)$ time [11]. It takes advantage of the fact that when adding the ith sequence in stepwise-addition, the score of the tree with $i-1$ taxa has already been calculated. This reduces the overall complexity of GREEDY to $O(n^2m)$. The Three-Way Labels method can also be applied to TBR, reducing its complexity from $O(nm)$ to $O(m)$. The software written for this study used the Three-Way Labels method as well as other techniques for improving the performance of TBR [15,16]. In addition, it took advantage of the algorithms for rapidly calculating upper and lower bounds and the exact values of the Hamming distances between sequences, yielding a several-fold improvement in performance [17]. As a result, the overall speed of the software was comparable to PAUP*.

6. Conclusions

The experiments described here demonstrate that varying representation of the search space, independently of search regime, can qualitatively affect the performance of stochastic search procedures for phylogenetic tree reconstruction. In particular, hill-climbing with the TBR operator over the direct tree representation runs like a sprinter, with large improvements to the tree score with a few iterations, then rapidly slows down. In contrast, GHC hill-climbing over the greedy decoder representation performs like a long-distance runner, gradually improving the tree score even after a long period of time. The differing advantages of both can be combined by using GHC initially and feeding its result to TBR; these hybrid algorithms perform by far the best. The most effective combination partitions an allocated run time so that the majority is spent in GHC, with a fast burst of TBR at the end. The best of the algorithms reported here significantly outperform PAUP*, the most popular software program for phylogenetic tree reconstruction.

The stochastic algorithms that this study describes use a simple hill-climbing method of searching through a solution space. Other techniques such as parallel hill-climbing, simulated annealing, and genetic algorithms have shown improved performance for other types of problems [14]. The greedy decoder representation is just one example of an alternative to the direct representation for phylogenetic trees. Other possibilities, perhaps based on those proposed for the combinatorial optimization problems of number partitioning [14] or graph bisection [18], may yield further benefit, as they have for those problems.

Finally, there have been several recently developed algorithms that build on TBR and have showed marked improvement in performance [19]. Some, for example, take a divide-and-conquer approach, where the taxa are partitioned into smaller stochastic optimization problems, then combined into a larger tree [20]. The Ratchet technique randomly adds weights to a subset of characters in the sequences before branch swapping is performed [21]. In this study, we have shown that hybrid algorithms that begin with GHC can perform better than TBR alone. Thus, the performance of any of these new TBR-based algorithms could potentially be improved by combining them with GHC selection of an initial tree.

Acknowledgment

This work was funded in part by Grant T32-HG02295 from the NIH.

References

[1] Durbin R, Eddy S, Krogh A, Mitchison G. Biological sequence analysis. Cambridge, UK: Cambridge University Press; 1998.
[2] Rice K., Warnow T. Parsimony is hard to beat. In: Proceedings of COCOON, 1997.
[3] Felsenstein J. Evolutionary trees from DNA sequences: a maximum likelihood approach. J Mol Evol 1981;17(2):368–76.
[4] Mount D. Bioinformatics: sequence and genome analysis. Cold Spring Harbor, New York: Cold Spring Harbor Laboratory Press; 2001.
[5] Kim J., Warnow T. Tutorial on phylogenetic tree estimation, 1999.
[6] Salter L. Algorithms for phylogenetic tree reconstruction. In Proceedings of the International Conference on Mathematics and Engineering Techniques in Medicine and Biological Sciences, 2002, pp. 459–465.
[7] Swofford D. PAUP*, phylogenetic analysis using parsimony (and other methods), 2003.
[8] Lundy M. Applications of the annealing algorithm to combinatorial problems in statistics. Biometrika 1985;72(1):191–8.
[9] Lundy M, Mees A. Convergence of an annealing algorithm. Math Program 1986;34:111–24.

[10] Dress A, Kruger M. Parsimonious phylogenetic trees in metric spaces and simulated annealing. Adv Appl Math 1987;8:8–37.
[11] Ganapathy G., et al. Better hill-climbing searches for parsimony. In: 3rd Workshop on Algorithms in Bioinformatics, 2003.
[12] Maddison D. The discovery and importance of multiple islands of most-parsimonious trees. Syst Zool 1991;40(3):315–28.
[13] Rice K et al. Analyzing large data sets: rbcL500 revisited. Syst Biol 1997;46(3):554–63.
[14] Ruml W, Ngo J, Marks J, Shieber S. Easily searched encodings for number partitioning. J Optimiz Theory Appl 1996;89(2):251–91.
[15] Goloboff P. Methods for Faster Parsimony Analysis. Cladistics 1996;12:199–220.
[16] Ronquist F. Fast fitch-parsimony algorithms for large data sets. Cladistics 1998;14:387–400.
[17] Weber G. Data Representation and Algorithms for Biomedical Informatics Applications. PhD thesis, Harvard University, Cambridge, MA, 2005.
[18] Marks J, Ruml W, Shieber S, Ngo J. A seed-growth heuristic for graph bisection. In: Battiti R, Bertossi AA, editors. Proceedings of Algorithms and Experiments (ALEX98). Italy: Trento; 1998. p. 76–87.
[19] Goloboff P. Analyzing large data sets in reasonable times: solutions for composite optima. Cladistics 1999;15(4):415–28.
[20] Huson D, Nettles S, Parida L, Warnow T, Yooseph S. The disk-covering method for tree reconstruction. In: Battiti R, Bertossi AA, editors. Proceedings of Algorithms and Experiments (ALEX98). Italy: Trento; 1998. p. 62–75.
[21] Nixon K. The parsimony Ratchet, a new method for rapid parsimony analysis. Cladistics 1999;15(4):407–14.