Methods of Phylogenetic Analysis: New Improvements on Old Methods.

Biochem 218 Final Project
Michael Gordon

March 10, 2003

Introduction:

Phylogenetic analysis presents a unique problem in biology, because evolutionary history can *never* be known with certainty. The purpose of any phylogenetic analysis is to estimate the evolutionary relationships between a set of homologous taxa, which can be anything from morphological characteristics to molecular sequences (reviewed in Mount, 2001). The result is a tree composed of "nodes" and "branches", where the terminal nodes (or "leaves") correspond to the taxa being studied (henceforth assumed to be DNA sequences), the internal nodes represent ancestral sequences, and the branches represent the topological relationship between the nodes (reviewed in Saitou, 1996).

While many different methods have been developed for inferring phylogeny, they can all be thought of as members of two broad classifications: (1) those methods that use an algorithm to directly build a tree through a series of defined steps; and (2) those methods that define a criterion to be maximized (or minimized), and then use an algorithm to evaluate potential trees based on this criterion (Swofford et al., 1996). The first class of programs works largely by converting the similarity between pairs of sequences into evolutionary distance, and then using a defined set of steps to build a tree. These methods are computationally very fast, but suffer from two major downfalls: (1) evolutionary information is lost when overall similarity between sequences is observed, rather than individual mutation events (Hendy and Penny, 1982); and (2) it is difficult to reliably assess the confidence of a tree produced by any given algorithm (Swofford et al., 1996). For these reasons, purely algorithmic methods will not be discussed further.

In contrast, the second group of methods proceeds in two steps. First, an optimality criterion is defined, which is simply a score used to assess the value of a particular tree. Second, an algorithm is used to compute the value of this function for various trees, while searching for the best tree (the one that maximizes the criterion) (Swofford et al., 1996). While these methods are appealing because they have the promise of finding the optimal tree according to the applied criterion, they can be computationally slow for even moderate numbers of taxa, to a point where the amount of time required for an exhaustive search is prohibitive. However, this limitation has led to the development of countless computational methods that attempt to reliably get as close to the optimal tree as possible, in a reasonable amount of computational time.

In this paper, I will first briefly discuss the two most widely used criteria-based methods: Maximum parsimony and Maximum likelihood (ML). After outlining the theoretical and practical differences between these two methods, I will discuss recent attempts to improve the utility of maximum likelihood by using genetic algorithms to overcome the incredible computational power required to implement this approach.

Maximum Parsimony:

Maximum parsimony (or simply parsimony) analysis is intuitively appealing because it is based on finding the simplest solution to an observed set of data (reviewed in Swofford et al., 1996; Saitou, 1996). Essentially, parsimony attempts to build a tree that minimizes the number of evolutionary changes required to explain the observed data (in our case, DNA sequences). Therefore, the optimality criterion (which in this case is minimized) is total tree length. The length of the tree is defined as the number of character state transformations (mutations to the biologist) required to explain the

existence of the nucleotides at all positions in a set of aligned sequences (Swofford et al., 1996). Mathematically, this translates to finding a tree that minimizes the function:

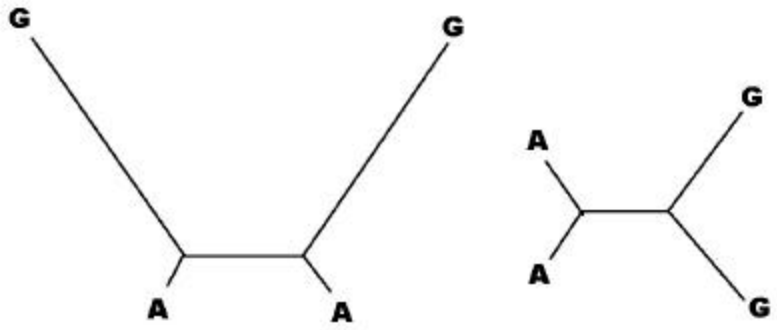
$$L(\tau)= \sum_{k=1}^{B} \sum_{j=1}^{N} w_j..\text{diff}(x_{k'j}, x_{k''j}) \qquad \text{(Swofford et al., 1996)}$$

Where $L(\tau)$ is the tree length; B is the number of branches; N is the number of nucleotide characters; k' and k'' are the two nodes incident to branch k; $x_{k'j}$ and $x_{k''j}$ are nucleotide characters from the data, or inferred characters on internal nodes (more on this below); and diff(y,z) specifies the cost of a transition from state y to state z (Swofford et al., 1996).

In simplified terms, parsimony algorithms evaluate a given tree by individually looking at each column of nucleotide characters in a set of aligned sequences, summing the length of the tree required to account for each of the characters in that column, and then summing the results for all columns (Saitou, 1996). To evaluate the tree length required for each column, nucleotide characters are inferred for each internal node, and the cost of traveling along each branch is then evaluated (Swafford et al., 1996).

Problems with Parsimony:

The very thing that makes parsimony analysis so appealing (its simplicity), is also its most serious shortcoming. Simply put, if two sequences sharing a node both have a 'G' nucleotide at a given position, the most parsimonious model is that the ancestral sequence also had a 'G', and there were no changes along either of the branches descendant from that ancestral node. Unfortunately, this is not necessarily the case. For instance, consider the following example (adapted from Mount, 2001):



The true tree is shown on the left represents the true phylogeny of four taxa, with their corresponding nucleotides at a given position. This tree illustrates a case of convergent evolution, where each 'G' arose independently along a branch with a higher rate of sequence evolution than those leading to the 'A' characters. However, the most parsimonious tree that explains the data is shown on the right. This is a simple case where parsimony would not find the correct tree due to long branch lengths leading to the potential for super-imposed, unobserved mutations.

More complicated proofs of this failure of parsimony analysis have been presented previously (for example, Hendy and Penny, 1989 and Felsentstein, 1978). However, each boils down to the fact that parsimony is only reliable when the sequences

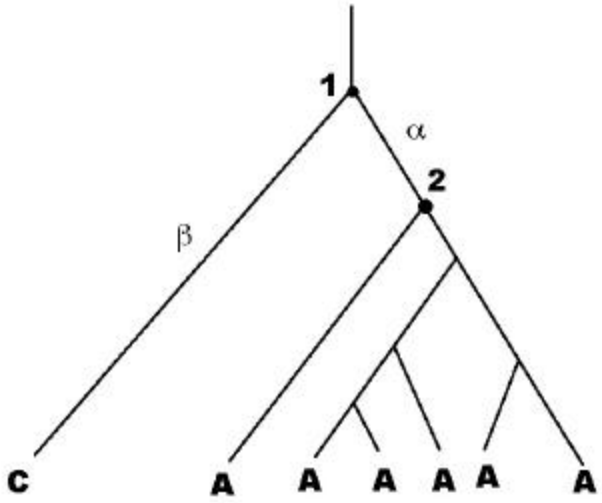
being compared are very closely related. Any time there is variation in the evolutionary rate among branches of a tree, or a given branch represents enough evolutionary time to introduce a significant number of super-imposed mutations, parsimony will be unreliable. Essentially, parsimony would work perfectly if we could observe each mutation in a sequence over evolutionary time. Since this is impossible, we must look for a method that attempts to model the changes that occur during evolution by accounting for unobserved as well as observed mutations.

A related problem with parsimony is that only certain positions in the sequence alignment yield information. For a position to be informative, it must have at least two different examples of at least two different characters (Mount et al., 2001). Therefore, positions that do not meet these criteria are deemed uninformative, even though under a different evolutionary model (for example maximum likelihood), information could be gleaned from these positions.

Another potential problem with parsimony is that it is significantly slower than the purely algorithmic distance-based methods mentioned above. This is due to the potentially enormous number of trees that must be evaluated, depending on the number of taxa involved. In fact, the number of unrooted bifurcating trees for *n* taxa is given by: $N=(2n-5)!/[2^{n-3}(n-3)!]$ (Saitou, 1996). This means that for 20 taxa, there are 2.2×10^{20} possible unrooted trees. However, this problem has largely been overcome for parsimony by using more sophisticated tree-searching algorithms, some of which will be discussed in a later section with respect to their application in Maximum likelihood analysis (which is orders of magnitude more computationally intense than parsimony (Lewis, 1998)).

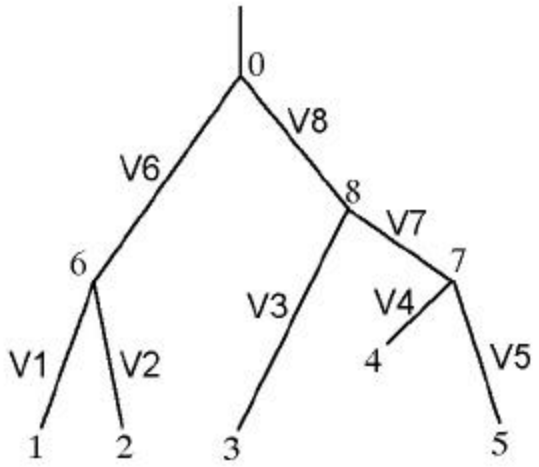
Maximum Likelihood, Part I: The optimality criterion

While parsimony methods seek phylogenetic solutions that minimize the amount of evolutionary change required to explain a data set, Maximum likelihood methods attempt to find solutions that have a maximum probability of being correct, given a particular evolutionary model (Swofford et al., 1996). This distinction may at first appear semantic, but it is extremely important when (as described above) the evolutionary time involved is long enough to produce a substantial number of multiply mutated positions within a sequence. Furthermore, unlike parsimony, maximum likelihood methods consider branch lengths when calculating the probability of a particular tree being correct. The importance of this difference is illustrated in the following example (adapted from Swofford, 1996):



Shown above is a theoretical "true" tree, and the nucleotides at a given position within each taxa. α represents the branch from node 1 to node 2, and β represents the branch from node 1 to the leaf with nucleotide 'C'. Assuming that node 2 represents an ancestral 'A', we know that a mutation must have occurred somewhere along α or β. While a parsimony method would find equal value in assigning an 'A' or a 'C' to node 1, intuitively we can see that this ancestor is more likely to be an 'A' because α is much shorter than β. This becomes critical if one was to add one extra branch, which terminated in a 'C'. Under parsimony, this branch could be added onto either α or β with an equal number of evolutionary changes. However, maximum likelihood would favor a tree where the new branch was added to β, which is indeed the more likely scenario.

Much like parsimony, maximum likelihood evaluates each column in a multiple sequence alignment independently (these methods are sometimes referred to as "character based"). Therefore, if we consider only one column, maximum likelihood evaluates a given tree based on the total probability that each of its branches exists. In other words, what is the probability that a change occurs between characters *x* and *y* along the length of the branch that connects them? Since a Markov model is assumed, these individual probabilities are considered independent, and can be multiplied to produce the total probability for a given column. Furthermore, since each sequence position (column) is assumed to be independent, the total probability for a tree is the product of all the column probabilities (Felsenstein, 1981; Swofford et al., 1996). Mathematically, this was presented by Felsentstein (1981) as follows:



For the tree shown above, numbers 0-8 represent the character states (nucleotides) at each node in the tree. Clearly, some of these states (1-5) are known, while others (0,6,7,8) are unknown. Let s_x represent state *x* and v_z represent branch *z*. Therefore, $Ps_xs_y(v_z)$ represents the probability of changing from *x* to *y* over branch length *z*. The prior probability of s_0 (the root) is represented by πs_0, and is determined by the base composition of the sequences being studied. The likelihood of the tree would be:

$$L = p_{s_0} P_{s_0s_6}(v_6) P_{s_6s_1}(v_1) P_{s_6s_2}(v_2) P_{s_0s_8}(v_8) P_{s_8s_3}(v_3) P_{s_8s_7}(v_7) P_{s_7s_4}(v_4) P_{s_7s_5}(v_5)$$

However, states 0,6,7, and 8 are unknown, and rather than inferring them (like parsimony), maximum likelihood sums the probability of all possible bases in each of these positions, giving:

$$L = \sum_{s_0} p_{s_0} \{\sum_{s_6} P_{s_0s_6}(v_6)[P_{s_6s_1}(v_1)][P_{s_6s_2}(v_2)]\} \{\sum_{s_8} P_{s_0s_8}(v_8)[P_{s_8s_3}(v_3)][\sum_{s_7} P_{s_8s_7}(v_7) (P_{s_7s_4}(v_4))$$

$$(P_{s_7s_5}(v_5))]\}$$

This equation can then be made into algorithmic form if one assigns conditional probabilities to each node. Let $Ls_k(k)$ be the conditional likelihood for the state of node *k*. Then, as we traverse the tree from the leaves inward, for any node *k*, whose immediate decendants are *i* and *j*, we can compute the conditional likelihood for all four values of s_k as follows:

$$Ls_k(k) = (\sum_{s_i} P_{s_ks_i}(v_i) Ls_i(i)) (\sum_{s_j} P_{s_ks_j}(v_j) Ls_j(j))$$

This algorithm is iterated until it reaches node 0, where the overall likelihood is calculated by the following equation:

$$L = \sum_{s_0} p_{s_0} Ls_0(0)$$

The above equations begin to illustrate the computational intensity of this method, especially if the number of taxa becomes large. Not only does the number of trees to be evaluated quickly become prohibitive (as mentioned above), but evaluating each tree becomes increasingly laborious. This problem is exacerbated further by other functions required to optimize the branch length for *every* tree evaluated (not discussed explicitly here, but reviewed in Swafford et al., 1996).

Problems with the Maximum likelihood criterion:

Aside from the computational power required to implement this method (discussed in detail below), it is difficult to find much fault with maximum likelihood methods. This is largely because it can be broadly applied using whatever evolutionary model for sequence change over time that a specific implementation requires. However, if one looks at the assumptions that maximum likelihood algorithms make, there could be slight cause for concern. These assumptions are essentially the independence of changes over different branches of a tree, and of different positions within a sequence. The latter assumption is likely violated in a number of cases. These include: (1) the accumulation of insertions and deletions, each of which tend to involve more than one adjacent nucleotide within a sequence; and (2) selective pressure on protein coding stretches of DNA. One can easily imagine that mutation at a given site could put selective pressure on an adjacent site to change (or not to change), thereby yielding a more desirable codon. For instance, if a GTA codon (valine) is mutated to TTA (leucine), the protein might be fully functional, but a subsequent change in the third position to TTT (phenylalanine) would likely impair the protein's function and be selected against, while the same mutation in the original codon (GTA to GTT) would be silent. Some more recent efforts have been made to address this problem (see Goldman and Yang, 1994).

Another potential problem with maximum likelihood is that it does not appear to account for species to species changes in genome base composition. For instance, the overall base composition of the data set is incorporated into the π_{s_0} term described above, but this does not account for variability between regions of the tree. These variations could have an impact on the likelihoods of transition between specific nucleotides in local areas of the tree. It seems that local estimates of nucleotide composition could be incorporated into a maximum likelihood algorithm. In fact, this problem has been tackled by Galtier and Gouy (1995).

Others, including Gaut and Lewis (1995), have pointed to another source of problems with maximum likelihood: site to site variation in the rate of mutation. This problem, and the several solutions designed to account for it, are beyond the scope of this paper, and will not be discussed.

Despite the potential problems with the original implementation of ML (many of which have since been addressed to some extent), it is widely considered the best way to perform phylogenetic reconstruction, and has outperformed other methods on a number of occasions (eg. Kuhner and Felsenstein, 1994; Huelsenbeck, 1995). We will therefore assume that this is the current ideal, look at the practical limitations of its implementation, and examine how these limitations can be overcome.

Maximum Likelihood II: Finding the best tree (the old way).

Exhaustive approaches:

For small numbers of taxa (a popular threshold is 11), the simplest and best way to find the most likely tree is to evaluate all possible trees. Such an exhaustive search is guaranteed to find the most likely tree. Similarly, the branch-and-bound method developed by Hendy and Penny (1982) will find the best tree, but without doing an exhaustive search. The principle of the branch-and-bound method is to make what amounts to a tree of all possible trees (referred to here as the search tree). In other words, a tree with only three taxa is placed at the top of the search tree, and each successive branch of the search tree corresponds to adding a new branch to a particular existing branch of the phylogeny being built. The branch-and-bound method then takes a depth-first approach to traversing the search tree, while eliminating paths that cannot possibly lead to a phylogeny that is more likely than the best case tested so far (reviewed in Swafford et al., 1996).

Branch-and-bound methods are guaranteed to find the best possible tree topology, but are still quite slow. In fact, in the worst case scenario, all possible trees must still be evaluated. In today's age of whole organism sequencing, we routinely would like to be able to evaluate the phylogeny of gene families with many more members than can be handled by simple branch-and-bound methods. Therefore, we must look for less computationally intense ways to get as close to the optimal tree as possible.

Heuristic Approaches:

Heuristic approaches are those that attempt to follow the best path to a solution, based on information received along the way. These approaches can be very simple, or very sophisticated, and each comes with its own limitations.

When Felsenstein (1981) first presented his maximum likelihood principles, he identified the need for a search strategy that was "less ambitious" than an exhaustive search. Therefore in his implementation of ML, he employed what would now be referred to as a "stepwise addition" heuristic (reviewed in Swofford et al., 1996). Conceptually, this approach is similar to branch-and-bound methods. However instead of starting down every path of the search tree and eliminating paths only when it is impossible that the best solution lies farther along that path, stepwise addition follows only one path. This path is determined by starting with a tree of 3 taxa, determining the most likely place to add a fourth taxon, then taking the resulting four-taxon tree, finding the most likely place to add a fifth taxon, and so on.

While this approach is conceptually simple and appealing, it is also clearly very flawed. The flaw is that the globally optimal tree is unlikely to be the optimal choice at every addition step, and is therefore unlikely to be found. Stepwise addition is therefore referred to as a greedy algorithm (Swofford et al., 1996) – one that is always taking the best solution offered in the present, without ever looking into the future. In the worst case scenario, an early decision in the search tree will lead down a path with only final trees that are very far from optimal.

Due to the glaring problems with stepwise addition algorithms, attempts have been made to improve the resulting trees by performing rearrangements, such as "branch swapping". While the details of these analyses will not be discussed here, it is important

to note that methods do exist to improve trees made by even the most rudimentary heuristic algorithms. However, these methods also have serious limitations. Most notably, if a better tree lies several rearrangement steps away from the original tree, that solution will never be found if any of those steps do not (by themselves) provide an increase in likelihood.

Maximum Likelihood III: finding the best tree (a new way).

Genetic algorithms are not a new idea, but ironically have only recently been applied to biological problems (reviewed in Foster, 2001). The essence of a genetic algorithm is that it uses the principles of evolution to computationally arrive at a solution to an optimization problem. By selecting the fittest individuals among a set of randomly varied ones, over many generations, a genetic algorithm applies selective pressure on a population of solutions, hoping to eventually produce the fittest individual possible (Foster, 2001).

The terms used when describing genetic algorithms should not be confused with their biological meanings. For instance, each solution within a 'population' is often referred to as a 'chromosome' (or 'individual'). The measure of each chromosome's ability to solve the problem at hand is referred to as its 'fitness', and this fitness is defined by each individual's particular 'alleles', which correspond to the parameters to be optimized. In each 'generation', the fittest individuals leave the most 'offspring' to the next generation, after undergoing random events of 'mutation' and 'recombination' with other individual solutions (reviewed in Foster, 2001).

A genetic algorithm was first adapted for analyzing the phylogeny of nucleotide sequences by Lewis (1998). In the algorithm used there, the population in each generation consists of individual phylogenetic trees. The fitness of each tree is then assessed by calculating its natural log likelihood score (lnL). The parameters that define this score are: The tree topology; the individual branch lengths; the κ parameter (defined as the transition/transversion rate ratio by the HKY model (Hasegawa et al., 1985)); and the base frequencies (Lewis, 1998). It is important to note that in this algorithm, the branch lengths are varied along with the topology in each generation, but are not optimized for each tree. This is in contrast to all other implementations of ML, where the branch lengths are optimized prior to evaluating the likelihood of each tree examined (Lewis, 1998). This important distinction translates to significant time saved, as branch length optimization is the most computationally intensive part of evaluating each tree during traditional ML implementations (Lewis, 1998).

The genetic algorithm (Lewis, 1998) first defines a population of n individual tree solutions. In the first generation, these consist of random solutions to the defined problem. The fitness of each individual (defined by lnL) is computed as described above, and the individuals are ranked on the basis of this score. The value i is then defined by each individual's ranking, with i=1 corresponding to the highest ranked individual, and i=n being the lowest ranked individual. The probability of each individual leaving an offspring to the next generation is defined by $p(n - i + 1)$ (p is chosen such that the sum of all probabilities is 1), with the following exception: The individual with the highest lnL automatically leaves k offspring (k is a user-defined value). The rest of the next generation's population ($n - k$ individuals) is selected based on the probability defined above.

All of the offspring, except one copy of the fittest individual, are then potentially subjected to mutation and recombination. The excepted individual is left unchanged to ensure that the likelihood of the best solution in each successive generation cannot decrease. Mutations imposed on individuals can consist of changes in branch length, and/or topological rearrangements. Branches are selected for mutation with a probability defined by the user, and subjected to a multiplicative factor drawn from a gamma distribution. Similarly, the user also defines the proportion of topological rearrangements, each of which involves randomly removing a portion of the tree and reattaching it to a random position on the remaining tree (Lewis, 1998). Finally, recombination involves taking an offspring tree (which may or may not have been mutated), removing a portion of that tree, and attaching it to a different parental tree, which has been pruned to remove the corresponding leaves. Again, this occurs with a frequency that is defined by the user.

It is the recombination described above that sets genetic algorithms apart from other heuristic approaches to ML, because it allows for the potential of two good portions of two different trees being brought together (Lewis, 1998).

In the first test of the genetic algorithm described above (implemented in a program called GAML), a 55-taxon phylogeny was reconstructed 3 times, taking a total of 42.4 hours of CPU time (Lewis, 1998). The final solution was different in each case, with one of the 3 topologies corresponding to the best solution found by an implementation of PAUP* (a program based on stepwise addition followed by extensive branch swapping)(Swofford and Begle, 1993). However, PAUP* took 783.h to arrive at this solution, demonstrating the significant increase in speed afforded by the genetic algorithm.

The value of the Genetic Algorithm:

While the speed of Lewis' genetic algorithm is certainly impressive, the lack of convergence on a single solution by three trials is somewhat disconcerting. However, this result is not altogether surprising given the principles on which this method is based. Like evolution, the GA depends on the combination of random change and selection to produce an optimal solution. This is inherently appealing to the biologist, because we embrace the power of selecting random changes over time, and because of the lack of simplifying assumptions that will bias the outcome in a particular direction (Foster, 2001). However, as we know from biology, random selection does not always lead to the "best" solution to a given problem (see the vertebrate eye). Similarly, it appears that genetic algorithms will not always (or even often) arrive at the most likely solution to a phylogenetic problem.

More recent investigations of this method have revealed that different trials with the same data set gave populations of trees that varied significantly from each other throughout the analysis, despite only small amounts of variation within each population (Brauer et al., 2002). This once again suggests that different trials can explore very different subsets of the total search space, and may not converge on a single solution. In essence, this is a variation on the 'hill climbing' effect of greedy algorithms discussed above. Genetic algorithms appear to do a better job of selecting a good hill to climb (than, for instance, stepwise addition), but they are still susceptible to getting trapped in local optima.

The variation between populations did, however, offer an intriguing way to improve the performance of the GA. In one trial, mixing two populations after they had been allowed to evolve (and approach a steady state) independently, offered a substantial increase in the accuracy of the best solution found after only a small number of additional generations (Bauer et al., 2002). This could offer a way of increasing the total search space covered in a given trial, and decreasing the possibility of getting caught in a local optimum.

Another attempt to improve the performance of genetic algorithms in phylogenetic reconstruction has been presented by Lemmon and Milinkovitch (2002). Their approach also involves the separate evolution of several distinct populations, but instead of mixing these populations, information about the trees in each is shared between them (referred to as MetaGA). For instance, if several populations converge on a solution for one section of the tree, that section is henceforth left unmutated, based on the assumption that this part of the solution is correct. By progressively restricting the amount of each tree subjected to mutations and rearrangements, this algorithm offers a significant increase in speed (Lemmon and Milinkovitch, 2002). While very efficient, this technique intuitively appears more susceptible to local optima. The reason for this is simply that even if a region of a tree topology has a very high likelihood score, there is no guarantee that this topology will be consistent with that of the globally optimum tree. Therefore, by holding high scoring regions constant, we remove an element of the random selection that makes GA appealing, and add an element which is theoretically similar to quartet puzzling methods (Strimmer and von Haeseler, 1996). Given the mediocre performance reported for quartet puzzling, it seems unlikely that the metaGA would perform better than the standard GA approach. However, the comparative accuracy of these two methods has yet to be firmly established experimentally.

Final thoughts on GAs:

Are genetic algorithms a great new way to do phylogenetic reconstructions? The answer is likely dependent on exactly what the researcher is looking for. Essentially, every method's utility is judged on just two factors: speed and accuracy. In terms of speed, GAs perform extremely well. The first implementation of GA required 18-fold less computing time than the same reconstruction using PAUP* (Lewis, 1998). This is certainly not a trivial difference when it is translates to the difference between 2 days and well over a month. Additionally, recent attempts to parallelize GAs promise to reduce the time taken to do each analysis even further (Brauer et al., 2002).

In terms of accuracy, it appears that GAs perform well, but could be better. As discussed above, GAs have been found to be susceptible to local optima, and will randomly travel through very different search space on successive trials. Furthermore, attempts to optimize the user-defined variables for this method revealed that changes could have a dramatic effect on the resultant accuracy (Brauer et al., 2002). This suggests two things: (1) work needs to be done to optimize these variables; and (2) these methods appear susceptible to systematic error, since they can be biased significantly by a change in the variables. However, ultimately the solutions provided by the GAs were not far from those produced by other, more exhaustive methods, especially if the branch lengths of the final tree are optimized (Lewis, 1998; Brauer et al., 2002).

I believe that the most promising improvement that could be made to genetic algorithms involves the mixing of independent populations mentioned above. Data from trials of the GA indicate that even populations with a small number of individuals are capable of evolving good trees (Brauer et al., 2002). Therefore, one could potentially devise an algorithm that (for instance) started with four distinct populations of 5 individuals, let each population approach a steady state, then combine two pairs to make two populations of 10 individuals. Once again, these mixed populations would be allowed to evolve before combining them into one population of size 20. I believe that such an algorithm could add the benefits of population mixing (something that clearly occurs during biological evolution), without increasing the computational time too dramatically (since the total number of individuals would stay constant). Furthermore, one could imagine that such an algorithm would benefit from having the most dramatic changes occur in small populations, and the fine-tuning occur in larger populations. This could be an advantage because early in the evolution process, changes accumulate rapidly, and there is a high amount of variability between individuals in a population, suggesting that small numbers of individuals are sufficient. Later in the evolution process, variation within a population becomes much smaller, making large populations preferable at this stage.

Conclusions:

In today's age of rapid, genome scale sequencing, the average biologist can easily find himself with the prospect of analyzing the phylogenetic relationships between a large number of homologous sequences. It is therefore of great utility to devise methods that perform such a task well, and in a reasonable amount of time. Maximum likelihood approaches have been established as the gold standard for evaluating the quality of a tree, but the computing time required to find the optimal tree can be prohibitive. While there have long been methods that can make good predictions for large numbers of taxa in a matter of weeks, this time frame is still long enough to restrict access to only those with both expertise and fast computers that they are willing to dedicate to a particular job. In an ideal world, any biologist would be able to construct a good phylogeny for his or her favorite gene family, and be finished in a matter of hours (or perhaps overnight). I believe that the implementation of maximum likelihood criteria using a genetic algorithm combines the speed and accuracy required to approach this ideal.

References:

Brauer, M.J. et al., (2002). Genetic algorithms and parallel processing in maximum-likelihood phylogeny inference. *Molecular Biology and Evolution*, 19: 1717-1726.

Felsenstein, J. (1978) Cases in which parsimony and compatibility methods will be positively misleading. *Systematic Zoology*, 27: 401-410.

Felsenstein, J. (1981). Evolutionary trees from DNA sequences: A maximum likelihood approach. *Journal of Molecular Evolution*, 17: 388-376.

Foster, J.A. (2001). Evolutionary Computation. *Nature Reviews: Genetics*, 2: 428-436.
Galtier, N., and Gouy (1995). Inferring phylogenies from DNA sequences of unequal base compositions. *PNAS* 92: 11317-11321.

Gaut, B.S., and Lewis, P.O. (1995). Success of maximum likelihood phylogeny inference in the four-taxon case. *Molecular Biology and Evolution*, 12: 152-162.

Goldman, N. and Yang, Z. (1994). A codon-based model of nucleotide substitution for protein-coding DNA sequences. *Molecular Biology and Evolution*, 11: 725.

Hasegawa, M., Kishino, H., Yano, T. (1985) Dating of the human-spe plitting by a molecular clock of mitochondrial DNA. *Journal of Molecular Evolution*, 21: 160-174.

Hendy, M.D. and Penny, D. (1989). A framework for the quantitative study of evolutionary trees. *Systematic Zoology*, 38: 297-309.

Hendy, M.D., and Penny, D. (1982). Branch and bound algorithms to determine minimal evolutionary trees. *Mathematical Biosciences*, 59: 277-290.

Heulsenbeck, J.P. (1995). The robustness of two phylogenetic methods: Four-taxon simulations reveal a slight superiority of maximum likelihood over neighbor joining. *Molecular Biology and Evolution*, 12: 843-849.

Katoh, K. Kuma, K., Miyata, T. (2001). Genetic algorithm-based maximum-likelihood analysis for molecular phylogeny. *Journal of molecular evolution*, 53: 477-484.

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

Lemmon A.R., and Milinkovitch, M.C. (2002). The metapopulation genetic algorithm: An efficient solution for the problem of large phylogeny estimation. *PNAS*, 99: 10516-10521.

Lewis, P.O. (1998). A genetic algorithm for maximum-likelihood phylogeny inference using nucleotide sequence data. *Molecular Biology and Evolution*, 15: 277-283.

Mount, D.W. (2001). Phylogenetic prediction. In *Bioinformatics*, chap. 6, pp. 237-280. Cold Spring Harbor Laboratory Press, Cold Spring Harbor, New York.

Saitu, N. (1996). Reconstruction of gene trees from sequence data. *Methods in Enzymology*, 266: 427-449.

Stimmer, K., and von Haeseler, A. (1996). Quartet puzzling: a quartet maximum-likelihood method for reconstructing tree topologies. *Molecular Biology and Evolution*, 13: 964-969.

Swofford, D.L., Olsen, G.J., Waddell, P.J., and Hillis, D.M. (1996). Phylogenetic Inference. In *Molecular systematics*, 2nd edition, chap. 5, pp. 407-514. Sinauer and Associates, Sunderland, Massachusetts