or NP-hard is equivalent to a proof that it cannot be solved in polynomial time, and that our algorithms will run in exponential time, or worse.

The shortest Hamiltonian path problem is one of those that has been proven to be NP-hard. An associated decision problem (Yes or no — Is there a solution to this SHP that has length less than X?) is known to be NP-complete. So no algorithm that we can reasonably expect to discover can run in polynomial time. Of course that does not prevent us from discovering one that runs in exponential time but runs rather rapidly on moderate-sized cases. We might, for example, find one that runs in $0.0000001e^n$ seconds. It would run quickly on small and moderate-sized cases, but it too would ultimately be defeated by the exponential growth of execution time as the problem size grew.

Branch and bound methods

In spite of its being NP-hard, there are ways to considerably speed up the SHP. The simplest is branch and bound. We have already seen that we can search exhaustively by traversing the search tree of solutions. But we need not actually traverse all of it. As we go up the tree, building up a solution, we can keep track of the total length of that part of the solution so far. We also will be keeping track of the best solution found so far, and how long it is. Suppose that the best solution so far has length 2.932. As we go up a branch on the search tree, before we reach the end of the branch, we notice that the total length of this partial solution has reached 3.193. Any further points that we add to the solution can do nothing but increase that length. We therefore know that no solution in that subtree of the search tree can be any better. This is the "bound" in the branch and bound method. We can cut our losses by ceasing further movement into that subtree and backing out. If we have backed out when there are still a considerable number of points left to be added to the solution, we have saved a lot of work.

The result is an algorithm that branches (searching all parts of the search tree) but also uses its bound to greatly economize on the amount of work. Implementing this branch-and-bound search, we find that for the numerical example it does indeed arrive at the correct solution, and much faster than straight exhaustive search. It takes 0.46 seconds instead of 10.85, a better than 20-fold improvement.

Phylogenies: Despair and hope

Branch and bound has speeded up the solution greatly, but it has not actually escaped from the constraints of the NP-hardness proof. In fact, branch and bound algorithms too have a complexity that is exponential — it's just that they have improved the coefficient in front of the formula and maybe on the size of the exponent. (For example, they might in some case have computation time proportional to $e^{0.03n}$ instead of $e^{0.5n}$.)

The parsimony problem for nucleotide sequences is one of a number of phylogeny problems that are known to be NP-hard. (Finding the best tree or trees is NP-hard, knowing what is the number of changes on the best tree is NP-complete.) The proof of these was given by Foulds and Graham (1982; Graham and Foulds, 1982). These phylogeny problems are examples of finding a Steiner tree in a graph. The set of all sequences is a graph, where adjacent points are connected if the sequences differ at one site. A *Steiner tree* is a tree of minimal length connecting a given set of points in a graph. Many Steiner tree problems are known to be NP-complete or NP-hard. (Generally, the problem of knowing the length of the tree is NP-complete, and the problem of finding the tree is NP-hard.) W. H. E. Day and co-workers have provided NP-completeness proofs for a variety of phylogeny criteria, most of which we introduce in later chapters. These include Wagner parsimony on a linear scale (Day, 1983), Camin-Sokal and Dollo parsimony (Day, Johnson, and Sankoff, 1986), compatibility (Day and Sankoff, 1986), least squares distance matrix methods (Day, 1986), a variant on the minimum evolution distance matrix method (Day, 1983), and polymorphism parsimony (Day and Sankoff, 1987).

There would seem to be reason for pessimism. But it is important to recall that exponential run time is not necessarily typical. The NP-hardness proof shows only that, given that no algorithm achieves polynomial time, for any problem size there are instances of it that will take exponential time. *But* these need not be biologically reasonable cases. The worst-case complexity of the problem is exponential. But what is the biological-average-case complexity?

In fact, it seems that some NP-hard problems (such as finding trees by compatibility, a method we consider later in this book) are very rapidly solved by branch and bound methods for typical biological cases. Other problems (such as parsimony) do not have such fortunate behavior.

Branch and bound for parsimony

The use of branch and bound algorithms to speed up exhaustive search for most parsimonious trees is closely analogous to the algorithm that we have just described for the shortest Hamiltonian path problem. The search tree is the tree of trees that we have already described in Chapter 4 (see also Figure 3.3). It is the tree of possibilities that results from adding the species to a tree in their numerical order, at each stage choosing one of the possible places to add that species. Thus we start with species 1 and 2 in a two-species tree, add species 3 in one of the 3 possible places, then add species 4 in one of the 5 possible places, and so on. Figure 5.3 shows this tree of trees, for a five-species case where the species are labeled A, B, C, D, and E. There are 15 possible tips, the 15 bifurcating trees, plus the interior nodes of the search trees which are 8 other incomplete trees.

We can imagine traversing this search tree. At each point on it, we have a partial (or a complete) tree. We can evaluate the number of changes that this tree requires on our data. This could be used in a branch and bound method, as was done in the SHP example. In their paper introducing the branch and bound method for phylogenies, Hendy and Penny (1982) have made some useful suggestions for

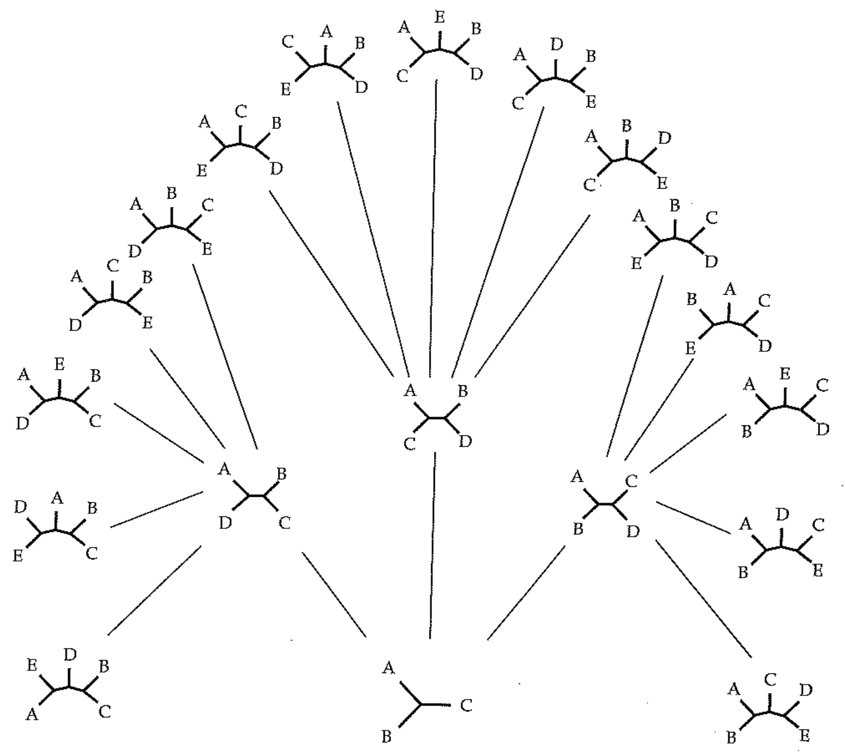

Figure 5.3: Search tree for most parsimonious tree in a five-species case.

improving the bound. Figure 5.3 shows the search tree, with all 15 unrooted bifurcating trees for 5 species. These are tied together by interior nodes that show all 3 four-species trees, and at the root is the single possible three-species tree. Figure 5.4 shows the same search tree with the trees themselves replaced by the number of changes of state that they require for the data in Table 1.1. The branch-and-bound traversal starts from the bottom of the search tree. In order to rule out as many trees as possible, as quickly as possible, it is helpful to find good trees soon. One strategy would be to search the nodes of the next level in the tree in order of the number of changes that their trees require. So we start at the bottom node (which requires 5 changes). At the next level we have nodes that require 8, 7, and 9 changes, respectively. If we make a preliminary visit to all three of them and

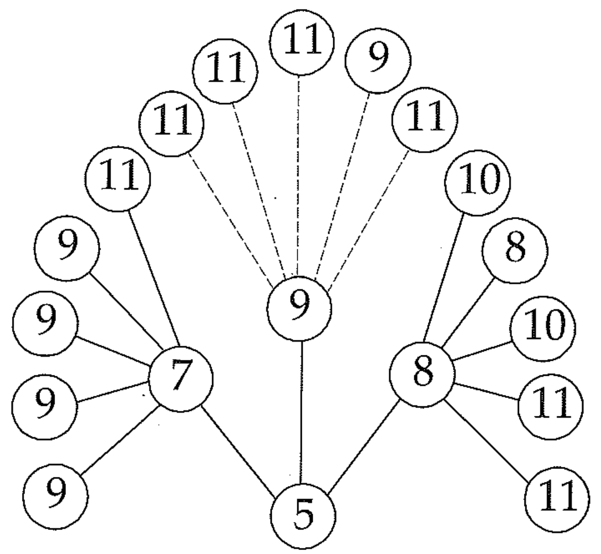

Figure 5.4: Search tree for most parsimonious tree for five species, using the data of Table 1.1. Trees are shown in Figure 5.3. Dashed lines are those not traversed by a branch and bound method. The species names in the data set correspond to labels A through E in Figure 5.3.

discover this, then we can plan to traverse the tree starting with the one that requires 7 changes. Proceeding up to it, we discover at the next level that there are 5 five-species trees, requiring 9, 9, 9, 9, and 11 changes, respectively.

Now we have candidate trees (the ones requiring 9 changes). We will be interested in any region of the tree whose bound is 9 or less. We will be uninterested in searching any region of the search tree that has all of its members requiring more than 9 changes. We proceed on to the next subtree, the one whose interior node requires 8 changes, so that its bound is 8. This has 5 five-species trees attached to it, and those require 10, 8, 10, 11, and 11 changes. Now we have a new candidate tree, requiring only 8 changes (and we discard the earlier ones that required 9). We are now interested in bounds of 8 or less. Finally, we start to examine the last of the three subtrees, whose interior node requires 9 changes. Its bound is 9. Immediately we know that none of the 5 trees attached to that interior node are of interest. All must require at least 9 changes, and we have already found a tree that requires only 8 changes. Hence we never travel along the branches of the search tree that lead beyond there (and they are therefore shown in Figure 5.4 as dashed lines). We are done, having examined only 10 of the 15 possible five-species trees.

The saving is not great in this example, but it can become enormous in larger cases. The saving is greater the less homoplasy there is in the data. In cases in which there are many conflicts between information from different characters and much parallelism and convergence, the branch-and-bound strategy does not perform particularly well.

Improving the bound

In the search tree of Figure 5.4, the bound is calculated simply by asking how many changes the partial tree at that node requires. This is a lower bound in the sense that it cannot be higher than the number of changes on any of the trees found farther out in the search tree. If we have found full trees that have as few as (say) 58 changes, then finding a partial tree that has 60 of them is sufficient reason to stop there and back out of that part of the search tree. None of the trees beyond that partial tree can have less than 60 changes, so none are candidates for being most parsimonious trees. We would like to calculate this lower bound on the number of changes so that it is as large as possible, and thus eliminate subtrees of the search tree as soon as we can, saving effort. There are further methods that help do this.

Using still-absent states

In many cases, we will be examining an interior node of the search tree corresponding to a partially constructed tree. Suppose that this tree has species A, B, D, and F on it. But species C and E have not yet been added to the tree. Suppose that the partial tree requires 48 changes. This will come from some of the characters that vary among species A, B, D, and F. But some of the characters will not vary until species C and E are added. We may be able to look at those species and see that, after they are added, there will be at least 11 more characters varying. In that case, no matter where they are added to the tree, the bound will be at least $48 + 11 = 59$. We can thus improve the bound considerably.

If we are dealing with 0/1 characters, that calculation is correct, but if the characters have multiple states, the bound can be made better by taking the multiple states into account. If a character has two states among species A, B, D, and F, but two more among C and E, then adding it will increase the number of changes by at least 2, not 1. Thus what we want to add to calculate the bound is the number of absent states, summed over all characters. This method of improving the bound is based on the paper by Foulds, Hendy, and Penny (1979). It has long been in use in branch and bound programs for inferring phylogenies, but this use was not described in print until the paper by Purdom et al. (2000).

Using compatibility

Another method of increasing the bound is to use not only the states in the individual characters but also the conflict between different characters. For two-state (0/1) characters, one can easily judge whether or not they can both have evolved on the same phylogeny with only one change each. We will cover this in more detail in Chapter 8. For now, we need only note that the two characters are *compatible* if they can evolve on the same phylogeny with only one change each, and that there is a simple test for this. If among all the species, all four of the combinations of states $(0,0)$, $(0,1)$, $(1,0)$, and $(1,1)$ are found, the characters are not compatible. If three or fewer are found, they are compatible.

That means that, when we consider the characters that are not yet varying on our partial tree, we can improve our lower bound on the number of changes of state. If we add all species to that partial tree, and in doing so now have variation in two incompatible characters that did not vary before, those characters must bring at least 3 changes of state with them. Each character individually will require one more change of state, and the pair will conflict, which means that one of them must have at least one additional change of state. If there are disjoint pairs of incompatible characters, each pair must bring with it 3 changes of state.

This method of computing the lower bound was developed by Foulds, Hendy and Penny (1979; see also Hendy, Foulds, and Penny, 1980). It was soon after applied to speeding up branch and bound methods, but the application to branch and bound search for most parsimonious phylogenies was first described by Purdom et al. (2000). If the species that remain to be added have k pairs of characters that are incompatible, and that do not now vary among the species on our partial tree, we must add $3k$ changes to the bound. Organizing the characters into pairs so that k is as large as possible can be done fairly quickly.

Increasing the bound as much as possible is important in getting a branch and bound method to run quickly. Hendy and Penny (1982) discovered that order of species was important, in particular that the most different species should be added as soon as possible. Purdom et al. (2000) describe improvements in speed by continually re-evaluating the order of addition during the search. Penny and Hendy (1987) describe a different branch and bound algorithm that adds characters one at a time rather than species.

Rules limiting the search

Another approach that has considerable promise is to rule out regions of the search tree in advance. Estabrook (1968) gave a rule which constrained the ancestral characters for the particular case of Camin-Sokal parsimony, a parsimony method that will be explained in Chapter 7. This might be used to speed branch and bound search. Estabrook's rule was rediscovered by Nastansky, Selkow, and Stewart (1973). They later (1974) presented an improved method that restricted the search further. However, these methods cannot be used with more general types of parsimony.

Andrey Zharkikh (1977; see also Ratner et al., 1995) has discovered some intriguing rules that allow us to determine that certain groups must be on all most parsimonious trees. Using them, we can reduce the size of the branch and bound