

Fast character optimization in parsimony phylogeny reconstruction

Mi Yan* David A. Bader†

University of New Mexico
Albuquerque, NM 87131 USA

August 27, 2003

Abstract

The problem of finding a phylogeny with maximum parsimony is one of the main problems in computational biology. While it is impossible to search the possible tree space exhaustively for large data sets, most heuristic approaches try to search in the neighborhood of sub-optimal trees. The speed of computing a score for each tree (e.g. tree length or total number of character changes) is as important as the different tree search strategies. Some techniques include short-cuts that have not been proven to be exact, and an incremental character optimization approach by which the speedup gains depends on data sets and is hardly analyzed in theory. This paper describes an exact and fast algorithm to compute tree length and our algorithm can obtain great speedup for any data sets. We discuss the algorithm for Fitch-parsimony, but it can also be applied to Wagner-parsimony.

*Department of Electrical and Computer Engineering, Email: miyan@ece.unm.edu

†Department of Electrical and Computer Engineering. Email: dbader@ece.unm.edu Phone: +1.505.277.6724, FAX: +1.505.277.1439.

1 Introduction

Phylogeny describes the evolutionary history of a set of species S and is usually modeled by an unrooted binary tree where the leaves are labeled by taxa in S and the internal nodes represent ancestral species. Phylogeny reconstruction is a major aspect of biological research. This is a very difficult computational problem because of the enormous tree topology space and often requires years to solve on some real data sets. In phylogenetic inference, many different estimation criteria for phylogeny reconstruction have been proposed [13]. The most common estimation criteria are Maximum Likelihood (ML) and Maximum Parsimony (MP). The principle of MP is to choose one tree which shows the smallest amount of evolutionary change. In the sequence based phylogeny, in which taxa are represented by aligned sequences, parsimony criteria are classified into Fitch Parsimony, Wagner Parsimony, Dollo Parsimony and Generalized Parsimony (Sankoff Parsimony) (see [13]).

Several algorithms of the heuristic search for MP trees are available ([7, 12]), but many of them are based on the same principle. In these algorithms, a sub-optimal MP tree is first constructed, then this tree is subjected to some kind of branch swapping to find a more parsimonious tree. Two of the most popular algorithms of branch swapping are (1) subtree pruning and regrafting (SPR), and (2) tree bisection and reconnection (TBR) ([12]). The heuristic algorithms are not guaranteed to obtain exact optimal solutions—the most parsimonious trees. A general approach applied to get the exact optimal is branch-and-bound (B&B) ([6]). The expansion of active nodes in the B&B search of phylogeny is stepwise addition, trying to insert the next taxon into arbitrary branches of an incomplete

tree. Many researchers have been working on different tree search strategies to escape local optima (e.g. [9, 8]) or prune branches more quickly ([10]). On the other hand, because parsimony problems require evaluating enormous number of trees, rapid evaluation of a candidate tree generated by stepwise addition or branch swapping is a crucial factor to the performance of a parsimony program as well.

The methods proposed by Farris [1] and Fitch [2] are still considered the basis of the calculation of tree length under Wagner parsimony and Fitch parsimony, respectively. Farris/Fitch methods require two passes: the first pass is sufficient to compute the minimum tree length and the second pass is required to make possible optimal state assignments to the internal nodes that lead to the minimum tree length. Obviously, those fast efficient programs (*Henning86* of Farris and *PAUP** of Swofford) use some special techniques for the optimization. However, the specific details on how to speed up searches in these codes are rarely discussed. Only a few published papers describe modifications of the Farris/Fitch methods that allow more efficient searches in two ways. One is multi-character optimization ([11, 8]), that packages multiple characters into a single memory word to optimize multiple characters simultaneously. The other technique compares the generated tree with the original tree and only updates the states of internal nodes when necessary ([4, 3]).

Goloboff ([4]) proposed the method for indirect calculation of tree lengths in 1993, which uses two passes and requires all the possible final states of the internal nodes to be found. For Fitch characters, this method assumes the final states of the potential root of each branch (X, Y) to be $S_X \cup S_Y$, where S_X and S_Y are the final states of node X and Y , respectively. During the SPR search, the procedure compares the state of the root of the clipped tree and

that of the potential root of the target tree, and if the intersection is empty, one is increased to the tree length. During the TBR search, a similar comparison is done between the states of the potential roots of two branches (one from the clipped tree and the other from the target tree).

Gladstein ([3]) proposed an algorithm for incremental optimization, based on preliminary state sets obtained from the first pass. The incremental algorithm updates the preliminary state sets from the point of reunion towards the root until the preliminary state sets of the new tree and that of original tree agree.

The method of Gladstein is exact, but the gain of this method depends on data sets and is difficult to analyze in theory. Basically we can say that it requires more time to derive the length of each rearrangement than the method of Goloboff. Goloboff stated that it is easy to prove that the method of indirect calculation of tree length will always produce the right length provided that the final states for the tree are correct ([5]). Generally, Goloboff's two-pass approach runs faster than Gladstein's incremental approach.

This paper presents an exact and fast algorithm based on the idea of the first pass of Fitch's method to compute the exact parsimonious tree length, that also requires two pass traversal the tree and is as fast as Goloboff's method and also proved to be correct. We describe our method under Fitch parsimony, but it is also applicable to Wagner Parsimony. The basic knowledge of Fitch's method and some local search techniques are explained in Section 2; Section 3 describes an exact and fast algorithm to compute tree length in local searches; and Section 4 draws the conclusions of our work.

2 Strategies for Parsimony

2.1 Fitch parsimony

In Fitch's parsimony, characters are unordered multistate characters. Character states can transform directly from one state to another state and back again. Based on the assumption that each character evolves independently, Fitch's method [2] examines only the evolution of one character on the tree at a time. Fitch's method requires two passes of a rooted binary tree. A state set is assigned to an internal node in the two pass optimization by combining information from some or all of the three surrounding nodes. The first pass of the optimization results in a *preliminary state set* and the minimum tree length whereas the second pass of the optimization gives the *final state set*. Starting with terminal nodes (leaves of the tree) having a single state set (the observed state) and initializing the tree length to be zero, the first pass proceeds from the leaves toward the root. The formulation of the *preliminary state set* of each internal node follows the following simple rule: the *preliminary state set* shall be the intersection of that of both children if the intersection is not empty; otherwise, it is the union of those sets and the tree length is increased by one.

A consequence of the reversibility of character states is that the tree may be rooted at any point with no change in the tree length. An unrooted binary tree can be rooted by adding a root node to any one of the branches. Thus, there is a *potential root node* for each branch in the tree. The potential root node is designated by R_{UV} , where U and V are the nodes adjacent to the root (see Figure 1A). Any potential root node can be chosen as *calculation root* for this purpose.

2.2 Local search strategies

Most heuristic algorithms require a local search in the neighborhood of a sub-optimal tree to find a more parsimonious tree. The most popular techniques includes the following.

- **Stepwise addition:** Some greedy algorithms and branch and bound algorithms add one taxa at a step. At each step, a new taxa is inserted into any branch of a given incomplete tree. Each arrangement results in a new tree topology and the cost of the new tree is required to be computed.
- **Subtree Pruning and Regrafting (SPR):** In the SPR approach, a branch of a tree is cut into two parts, a pruned subtree and residual tree. We call them the clipped tree and the target tree, respectively. The cutting point of the clipped tree is then grafted onto each branch of the target tree to produce a new topology.
- **Tree Bisection and Reconnection (TBR):** In the TBR approach, a tree is cut at a branch into two subtrees, one is the clipped tree and the other is the target tree. And these two trees are then reconnected by joining each pair of branches by a new branch.

3 An exact and fast algorithm to compute tree length

First let us see what happens during the local searches described in Section 2.2. As shown in Figure 1A, a taxa or a rooted subtree S is added to an arbitrary branch (U, V) of tree T to obtain a new tree T' . Assume the potential root on (R_{UV}, S) to be the calculation root

and one post-order pass of T' is sufficient to compute the parsimonious length of T' . The preliminary state set of R_{UV} will not change from T to T' . Our first goal is to preprocess T before the addition of taxa and compute the preliminary state set of R_{UV} for each branch (U, V) of T .

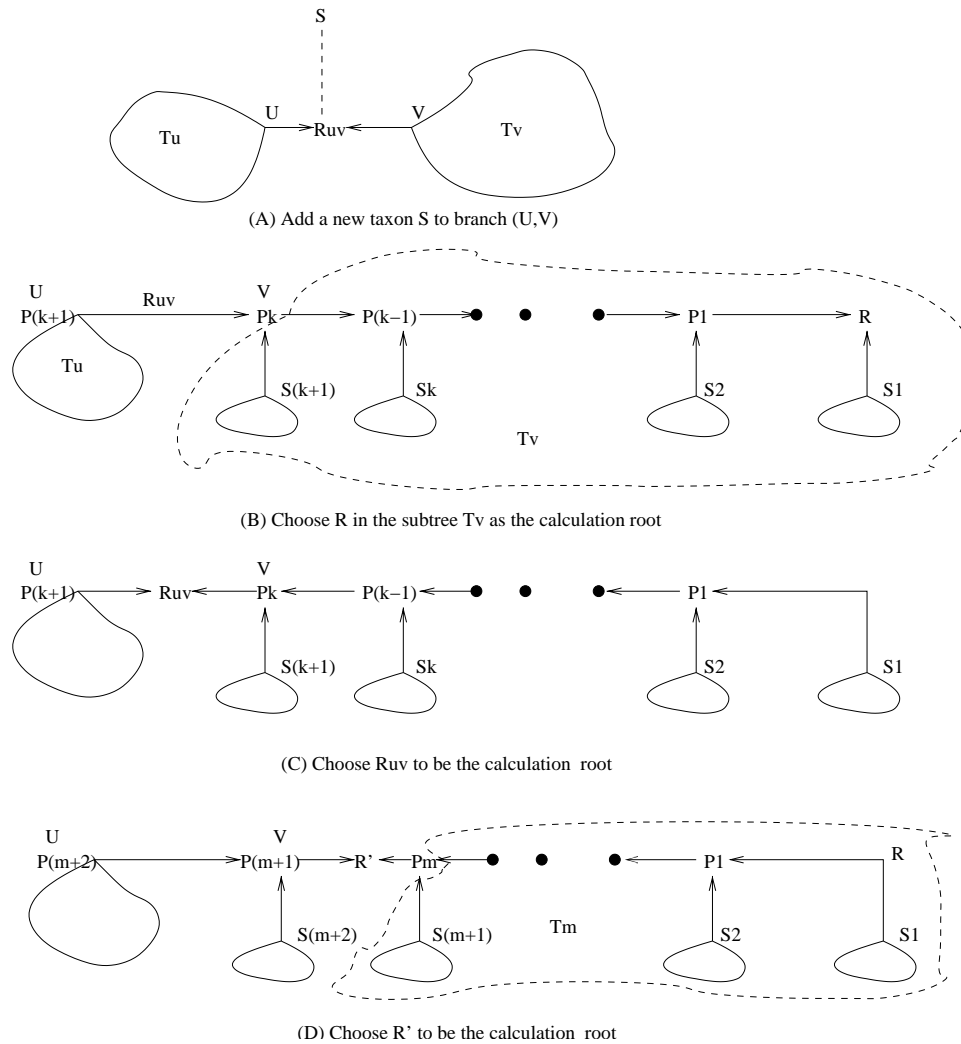


Figure 1: Choosing different potential root nodes as the calculation root

3.1 Computation of the final states of each potential root node

Without loss of general, let an arbitrary node R in the subtree T_V be the calculation root of T (see Figure 1B). In this rooted binary tree, V is the parent of U . Let P_{k+1} represent U and P_k represent V , then (P_k, \dots, P_1, R) is the path from P_{k+1} to R . P_{i-1} is the parent of P_i and S_i is the sibling of P_i for $i = 1, \dots, (k+1)$. In the post-order traversal of Fitch's method, the preliminary state set of node A comes from the information of all the nodes in the subtree rooted at A . In Figure 1C, we assume R_{UV} to be the calculation root. Comparing Figures 1B and 1C, one can see that the preliminary state sets of each node in $(P_{k+1}, S_{k+1}, \dots, S_1)$ will be the same whether R or R_{UV} is the calculation root since the subtree rooted at such a node does not change. But the preliminary state set of each node in (P_k, \dots, P_1) does change with the choice of calculation root and the preliminary state set of P_i depends on that of P_{i-1} and S_{i+1} (see Figure 1C). Therefore, we can first traverse the rooted tree in Figure 1B in post-order and get the preliminary state set of each node in $(P_{k+1}, S_{k+1}, \dots, S_1)$, next traverse the tree in pre-order and get the states sets of each node in (P_k, \dots, P_1) , and then obtain the state set of R_{UV} directly from that of P_{k+1} and P_k .

Assume an arbitrary root for the unrooted binary tree T , then for any branch (U, V) , we assume V is the parent of U without loss of generality. Use \overrightarrow{U} to designate the preliminary state set of the root of the subtree T_U which can be obtained from the the post-order traversal of T and use \overleftarrow{U} to designate the preliminary state set of the root of the subtree T_V which can be obtained from the pre-order traversal of T . Since characters are assumed to evolve independently, Algorithm 1 describes how two passes of the unrooted binary tree can produce

the final states of the potential root node on each branch for one single character.

```

Data :  $T$  rooted at  $R$ .
Result : (1)  $cost$ , the parsimonious length of  $T$ ; (2) The final state sets of the potential root on the branch between each node  $i$  and its parent.

begin
  (1)  $cost = 0$ ;
  (2) Traverse  $T$  in post-order.
  foreach internal node  $i$  of  $T$  do
     $\vec{i}$  = the intersection of the state sets of its two children;
    if  $\vec{i}$  is empty then
       $\vec{i}$  = the union of the state sets of its two children;
       $cost = cost + 1$ ;
    end
  end
  (3) Suppose  $R$  has two children  $lChild$  and  $rChild$ .
  (3.1)  $\overleftarrow{lChild} = \overrightarrow{rChild}$ ;
  (3.2)  $\overrightarrow{rChild} = \overleftarrow{lChild}$ ;
  (3.3) Traverse the subtrees rooted at  $lChild$  and  $rChild$  in pre-order, respectively.
  foreach node  $i$  whose parent is  $p$  and sibling is  $b$  do
     $\overleftarrow{i}$  = the intersection of  $\overleftarrow{p}$  and  $\overrightarrow{b}$ ;
    if  $\overleftarrow{i}$  is empty then
       $\overleftarrow{i}$  = the union of  $\overleftarrow{p}$  and  $\overrightarrow{b}$ ;
    end
  end
  (4) Compute the final state sets of the potential root nodes.
  foreach each branch between node  $i$  and its parent do
    Let the final state set of the potential root node to be the intersection of  $\overleftarrow{i}$  and  $\vec{i}$ ;
    if the intersection is empty then
      Let the final state set to be the union of  $\overleftarrow{i}$  and  $\vec{i}$ ;
    end
  end
end

```

Algorithm 1: Pre-process T to get the final states of each potential root node

Next we prove that Algorithm 1 always produces the correct final state sets of any potential root node R_{UV} . Use the pair (S_1, S_2) to represent the state sets obtained by Fitch's operation on state sets S_1 and S_2 . The proof uses induction as follows.

Proof

1. Suppose the calculation root to be R_{UV} . From step (4) of the algorithm, the final state of R_{UV} can be obtained both from the branch (U, R_{UV}) and the branch (V, R_{UV}) , which means the Fitch's operation on the pair $(\overleftarrow{U}, \overrightarrow{U})$ and the pair $(\overleftarrow{V}, \overrightarrow{V})$ should be equivalent. And from steps (3.1) and (3.2), both of them are equivalent to the Fitch's operation on the pair $(\overrightarrow{U}, \overrightarrow{V})$, which is the correct final state set of R_{UV} .

2. Suppose the calculation root R is located at the subtree T_V as shown in Figure 1B. Let k be the number of edges from V to R , then we have two cases, (a) $k = 1$ and (b) $k = m > 1$:

(a) When $k = 1$, then $V = P_1$ and $U = P_2$. From step (3.3), it is the case that $\overleftarrow{P}_1 = \overrightarrow{S}_1$ and $\overleftarrow{P}_2 = (\overleftarrow{P}_1, \overrightarrow{S}_2)$. Thus, $(\overrightarrow{U}, \overleftarrow{U}) = (\overrightarrow{U}, \overleftarrow{P}_2) = (\overrightarrow{U}, (\overrightarrow{S}_1, \overrightarrow{S}_2))$. The above value is just the final states of R_{UV} in Figure 1C.

(b) Assume the algorithm produces correct results when $k = m$. This implies that \overleftarrow{P}_{m+1} is the correct preliminary state set of the root of the subtree T_m in Figure 1D. From step (3.3), $\overleftarrow{P}_{m+2} = (\overleftarrow{P}_{m+1}, \overrightarrow{S}_{m+2})$. From step (4), $(\overrightarrow{U}, \overleftarrow{U}) = (\overrightarrow{U}, \overleftarrow{P}_{m+2}) = (\overrightarrow{U}, (\overleftarrow{P}_{m+1}, \overrightarrow{S}_{m+2}))$. The above value is equivalent to the final state of R_{UV} in Figure 1C.

For tree T with n leaves, step (2) of the algorithm labels $(n - 2)$ internal nodes; step (3) labels $(2n - 2)$ nodes; and step (4) labels $(2n - 2)$ branches. The algorithm in total takes $(4n - 6)$ Fitch's operations.

3.2 Computation of tree length in local search

When one searches the neighborhood of a tree by stepwise addition, SPR or TBR, we first use the approach in Section 3.1 to get the tree length of the original tree and the final state set for each potential root node, then for each rearrangement, only a constant time step is required to compute the increase of the tree length from the original tree to the new tree.

We explain next the details for different local searches.

- **Stepwise addition:** Add a new taxon S to tree T with n leaves. Preprocess T as demonstrated in Algorithm 1. Then for each new tree obtained by adding S into branch (U, V) , perform the Fitch's operation on (S, R_{UV}) to compute the increase of the tree length. The preprocessing needs $(4n - 6)$ Fitch's operations, for each of the $(2n - 3)$ new trees, and takes 1 Fitch's operation. Therefore, $(6n - 9)$ Fitch's operations are needed to calculate all the $(2n - 3)$ new trees, and the amortized Fitch's operations to evaluate each new tree is 3.
- **SPR:** For each SPR search, a tree T is split into a clipped tree T_c and a target tree T_t . In the preprocessing step, the first pass for T_c and both of the two passes for T_t are required. Let the basic tree length be the sum of the tree length of T_c and that of T_t . For each rearrangement by adding the root R_c of T_c into an arbitrary branch (U, V) of T_t , the increase of the tree length from the basic tree length can be obtained by doing Fitch's operation on (R_c, R_{UV}) .
- **TBR:** For each TBR search, a tree T is split into two subtrees T_c and T_t . In the preprocessing step, both passes of Algorithm 1 are required for T_c and T_t . Let the

basic tree length be the sum of the tree length of T_c and T_t . For each rearrangement by connecting branch (U, V) of T_c and (X, Y) of T_t , the increase of the tree length from the basic tree length can be obtained by doing Fitch's operation on (R_{UV}, R_{XY}) .

3.3 Experimental Results

As part of our work on searching all the most parsimonious tree by branch-and-bound approach (<http://www.unm.edu/~miyan/mp.src.1.tar>), we made some experiments to compare our method with the improved Fitch's method, which uses single pass optimization and for each new tree only the preliminary state sets of those nodes on the path from the point where the new taxon is inserted to the root of the original tree are recomputed. On SunBlade 100 with 500MHz USIII processor, we compare the running time of the branch-and-bound search and simulated data sets are used to study the performance of the two methods.

Some data sets used in the literature are based on ultrametric model tree and some test data are based on extreme evolutionary conditions. While the benchmark at <http://www.lirmm.fr/~ranwez/PHYLO/benchmarks24.html> include data sets more representative of one generally encountered by biologists and it allows comparison on trees whose internal branch lengths are not all equal, and over a wide variety of tree shapes and evolutionary rates. This benchmark includes data sets with 24 sequences of length 600 based on 2000 random 24-taxon trees and we used it to do the experiments. The speed improvement differs for different data set since the number of decomposed nodes in the branch-and-bound search varies. The more the decomposed nodes is, the faster our method runs. Generally, our method runs 2.3 - 3.5 times faster than the improved Fitch's method.

It is obvious that the more taxa a data set includes, the more advantage our method will gain compared with the improved Fitch's method. Limited to the enormous search space in the branch-and-bound search, the B&B search cannot work on large data sets practically. But the heuristic algorithms can apply to large data sets. For those algorithms that require computing tree lengths for all the trees in the neighborhood of a specific tree, our method will run much faster than the improved Fitch's method.

4 Conclusions

We presented an amortized constant time approach to compute the parsimonious tree length in stepwise addition, SPR and TBR, tree searches. Similar to Goloboff's method (see [4]), our method also requires two passes of a tree, then for each rearrangement, it needs a constant time to compute the difference between the tree length of the new tree and that of the original tree. Both of our two passes use the same rules as that of the first-pass of Fitch's method. Our algorithm has the same time complexity as Goloboff's and is faster than Gladstein's incremental method ([3]).

Furthermore, after the two-pass preprocessing of the original tree and the computation of the local change of the tree, we actually obtain the preliminary state sets of all internal nodes of the new tree supposing the point of reunion to be the calculation root. Therefore, if we save the preliminary state sets, when we search in the neighborhood of the new tree only one pass is needed during the preprocessing stage. Thus, during branch-and-bound search, only the tree associated with the root node in the search space requires two-pass

preprocessing and the following trees only require one pass preprocessing. While Globoff proposed some techniques to avoid the two passes during searching around the new tree, those techniques are difficult to implement and the consequent performance improvement is difficult to analyze.

Acknowledgments

This work was supported in part by NSF Grants CAREER ACI-00-93039, ITR ACI-00-81404, DEB-99-10123, ITR EIA-01-21377, Biocomplexity DEB-01-20709, and ITR EF/BIO 03-31654.

References

- [1] J. Farris. Methods for computing wagner trees. *Syst. Zool.*, 34:21–24, 1970.
- [2] W.M. Fitch. Toward defining the course of evolution: minimal change for a specific tree topology. *Syst. Zool.*, 20:406–416, 1971.
- [3] D. S. Gladstein. Efficient incremental character optimization. *Cladistics*, 13:21–26, 1997.
- [4] P. A. Goloboff. Character optimization and calculation of tree lengths. *Cladistics*, 9:433–436, 1993.
- [5] P. A. Goloboff. Methods for faster parsimony analysis. *Cladistics*, 12:199–220, 1996.
- [6] M.D. Hendy and D. Penny. Branch and bound algorithms to determine minimal evolutionary trees. *Math. Biosci.*, 59:277–290, 1982.
- [7] W.P. Maddison and D.R. Maddison. *MacClade: Analysis of phylogeny and character evolution*. Sinauer Associates, Sunderland, MA, 1992.
- [8] A. Moilanen. Simulated evolutionary optimization and local search: Introduction and application to tree search. *Cladistics*, 17:512–525, 2001.
- [9] K.C. Nixon. The parsimony ratchet : a new method fo rapid parsimony analysis. *Cladistics*, 15:407–414, 1999.

- [10] P.W. Jr. Purdom, P.G. Bradford, K. Tamura, and S. Kumar. Single column discrepancy and dynamic max-mini optimization for quickly finding the most parsimonious evolutionary trees. *Bioinformatics*, 2(16):140–151, 2000.
- [11] F. Ronquist. Fast fitch-parsimony algorithms for large data sets. *Cladistics*, 14(4):387–400, 1998.
- [12] D.L. Swofford and D.P. Begle. *PAUP:Phylogenetic analysis using parsimony*. Sinauer Associates, Sunderland, MA, 1993.
- [13] D.L. Swofford, G.J. Olsen, P.J. Waddell, and D.M. Hillis. Phylogenetic inference. In D.M. Hillis, C. Moritz, and B.K. Mable, editors, *Molecular Systematics*, pages 407–514. Sinauer, Sunderland, MA, 1996.