Molecular Phylogenetics and Evolution 79 (2014) 118–131

Contents lists available at ScienceDirect

Molecular Phylogenetics and Evolution

journal homepage: www.elsevier.com/locate/ympev

Hide and vanish: Data sets where the most parsimonious tree is known but hard to find, and their implications for tree search methods

Pablo A. Goloboff

Consejo Nacional de Investigaciones Científicas y Técnicas, Fundación Miguel Lillo, Miguel Lillo 251, 4000 S.M. de Tucumán, Argentina

ARTICLE INFO

Article history:
Received 9 March 2014
Revised 29 May 2014
Accepted 7 June 2014
Available online 18 June 2014

Keywords:
Parsimony
Tree searches
Homoplasy
Tree islands

ABSTRACT

Three different types of data sets, for which the uniquely most parsimonious tree can be known exactly but is hard to find with heuristic tree search methods, are studied. Tree searches are complicated more by the shape of the tree landscape (i.e. the distribution of homoplasy on different trees) than by the sheer abundance of homoplasy or character conflict. Data sets of Type 1 are those constructed by Radel et al. (2013). Data sets of Type 2 present a very rugged landscape, with narrow peaks and valleys, but relatively low amounts of homoplasy. For such a tree landscape, subjecting the trees to TBR and saving suboptimal trees produces much better results when the sequence of clipping for the tree branches is randomized instead of fixed. An unexpected finding for data sets of Types 1 and 2 is that starting a search from a random tree instead of a random addition sequence Wagner tree may increase the probability that the search finds the most parsimonious tree; a small artificial example where these probabilities can be calculated exactly is presented. Data sets of Type 3, the most difficult data sets studied here, comprise only congruent characters, and a single island with only one most parsimonious tree. Even if there is a single island, missing entries create a very flat landscape which is difficult to traverse with tree search algorithms because the number of equally parsimonious trees that need to be saved and swapped to effectively move around the plateaus is too large. Minor modifications of the parameters of tree drifting, ratchet, and sectorial searches allow travelling around these plateaus much more efficiently than saving and swapping large numbers of equally parsimonious trees with TBR. For these data sets, two new related criteria for selecting taxon addition sequences in Wagner trees (the "selected" and "informative" addition sequences) produce much better results than the standard random or closest addition sequences. These new methods for Wagner trees and for moving around plateaus can be useful when analyzing phylogenomic data sets formed by concatenation of genes with uneven taxon representation ("sparse" supermatrices), which are likely to present a tree landscape with extensive plateaus.

1. Introduction

Maximum parsimony and maximum likelihood are two criteria widely used in phylogenetic reconstruction. It has long been known that finding the most parsimonious or most likely tree is an NP-hard problem (Foulds and Graham, 1982). For data sets of some size, therefore, it is necessary to use heuristics, which rely on trial-and-error methods, such as trees created by successive addition of taxa and different types of rearrangement methods to explore tree space. The success of these search algorithms in the case of difficult data sets has been typically evaluated by considering the degree to which different methods provide similar answers, and/or by the best scores achievable by different methods for specific data sets.

In a recent paper, Radel et al. (2013) examined for the first time the ability of tree search algorithms to find the most parsimonious tree (MPT) in the case of data sets for which the uniquely MPT can be known exactly, but for which the characters have large amounts of homoplasy. Radel et al. (2013) reported that TNT (Goloboff et al., 2008a) succeeded in finding the MPT even in the largest cases examined (32,768 taxa). The data sets of Radel et al. (2013) are based on Chai and Housworth (2011); Chai and Housworth's method for constructing data sets is aimed at defining a unique MPT with a minimum number of binary characters; defining a unique tree with low numbers of binary characters requires having large amounts of homoplasy. These data sets, therefore, are not specifically designed for decreasing the chances of search methods to find the optimal tree and – despite the abundant homoplasy – they are in fact relatively tractable by standard search methods. For these data sets, TNT can find the MPT in relatively short times

E-mail address: pablogolo@yahoo.com.ar

http://dx.doi.org/10.1016/j.ympev.2014.06.008
1055-7903/© 2014 Elsevier Inc. All rights reserved.

with its more advanced search algorithms, but more standard algorithms are effective as well.

Creating data sets for which search methods have a low probability of finding the MPT requires considering the details of search methods, i.e. how random addition sequence (RAS) Wagner trees (Farris, 1970) and branch rearrangements operate. The present paper describes two other types of data sets for which finding the most parsimonious tree by standard search methods is very difficult – orders of magnitude more difficult than for Radel et al.'s (2013) data sets. While Radel et al. (2013) focused mostly on the mathematical aspects of the resulting data sets, the present paper concentrates more on the properties of tree searches. The difficulty of finding the most parsimonious tree by heuristic searches in the different types of data set is due to different reasons, and thus the solutions to find better trees are different for each type of data set.

The most difficult data sets studied here (see Section 5) have no homoplasy at all, but large proportions of missing entries – with observed entries allocated non-randomly. This non-random distribution of observed entries is likely to occur in phylogenomic data sets formed by concatenation (i.e. supermatrices). Thus, similar problems may arise for such concatenated matrices, and the search strategies proposed here may be helpful to find better trees in that setting as well.

2. Materials and methods

The number of taxa is denoted as **t**. The term "local search" is used for any search method (e.g. TBR, SPR, NNI; Swofford and Olsen, 1990) based on evaluating the trees in the "neighborhood" of the best solution(s) found so far. For branch-swapping, the TBR neighborhood of a tree **T** is the set of all trees that differ from **T** in just one TBR rearrangement (likewise for SPR or NNI). Note that if branch-swapping is defined in terms of the moves, then it is possible for two or more different moves to produce the same tree; thus, the number of possible TBR moves from **T** is different from the number of trees in the TBR neighborhood of **T** (although the difference between the two decreases with the number of taxa). The scripts used to create the data sets are available as Supplementary Material. The computers used to test the different search routines consist of a machine running under Windows 8, on an I7–3770 processor at 3.4 GHz, and a cluster of machines with somewhat slower processors (at 3.0 GHz) running under 64-bits Ubuntu Linux. The cluster was used to quickly replicate many searches, but each individual search (and thus all the timings) correspond to runs using a single processor. The probability of different outcomes of Wagner trees and TBR (reported under Sections 3.6, 4, and 4.3) was done using the C programs ***wagbias*** and ***tbrias***, available at <http://www.lillo.org.ar/phylogeny/published/Search_bias.zip> (Goloboff and Simmons, 2014).

3. Large amounts of homoplasy, easy landscape: Data sets of Type 1

This is the type of data set examined by Radel et al. (2013). They studied two cases, producing balanced (=symmetrical) and pectinate (=caterpillar) trees. As shown by Radel et al. (2013), TNT can easily find the MPT for these data sets. For comparable numbers of taxa, the search effort required to find the MPT is minimal, relative to the other types of data sets studied here.

3.1. Islands vs. local optima

Radel et al. (2013) only tested the ***xmult*** command, but in fact other search routines of TNT can also find the optimal tree easily. In this regard, Radel et al. (2013: 1189) posed that

"An interesting question for further theoretical work would be to determine whether or not the Chai–Housworth data allows non-optimal maximum parsimony trees that have locally optimal parsimony scores under tree re-arrangement operations. If local optimal trees do not exist, it would provide a basis to better understand the results reported here."

The question, however, should be formulated more strictly. A locally optimal tree can be defined as the individual (binary) tree for which the TBR neighborhood contains no trees of better score. The Chai–Housworth data sets have numerous local optima of this kind. This can be easily verified by using random trees (formed by sequentially adding each taxon at a randomly chosen branch of the growing subtree) as the starting point for TBR branch-swapping without saving multiple trees, and counting the cases in which swapping gets trapped at a suboptimal tree. For the symmetric case with **t** = 512 taxa, of 1000 random starting trees for TBR, 978 finished without reaching optimal score. For the pectinate case, 736 of 1000 random starting trees finished without reaching optimal score. Thus, trees locally optimal under TBR are common and numerous (although somewhat less so for the pectinate case).

But, in phylogenetics, it is common to talk about "islands" of trees, which were defined by Maddison (1991) in a different way: a TBR island is the set of all trees of similar score, which are separated by a single TBR rearrangement, or connected through trees of similar score which are in turn separated by a single TBR rearrangement. That a set of suboptimal trees forms an island cannot be confirmed unless all the trees of equal score that can be found via branch swapping starting from one of the trees in the set have already been saved and swapped – otherwise, saving one more additional tree could have led, by rearranging that additional tree, to a better tree. Saving multiple trees often leads to finding better results, and then the concept of "island" is more restrictive than the simple local optimum defined above. The Chai–Housworth data sets apparently do not have multiple islands of trees; when multiple trees are saved, branch-swapping from any starting tree invariably leads to the MPT, if enough trees are saved. The "enough" needs qualification here: saving only a few hundred trees suffices for essentially all starting points to lead to the MPT. For **t** = 512 and a balanced tree, saving as little as 250 trees, the MPT was found in every case, using 1000 different random trees as starting point. For **t** = 256 and a balanced tree, 10,000 such replications (with up to 500 trees saved) found the optimal tree in every single case. In the case of pectinate trees, the number of trees that need to be saved to find the MPT in every case is even lower; for **t** = 256, saving as little as 50 trees sufficed to find the MPT in every single case (out of 5000 replications). It is well-known that SPR is a much more superficial algorithm than TBR, and searches saving multiple trees under SPR are apparently also capable of finding the MPT for these data sets (both for the balanced and pectinate cases), but only when saving (and swapping) very large numbers of trees (even 20,000 trees are often insufficient for the balanced case with **t** = 256).

3.2. Tree landscape

A diagram representing the tree landscape for these data sets is shown in Fig. 1 (top curve). There is a single peak; flat regions where TBR may get swamped if not saving multiple trees exist, but are not too extensive. Given that they are not extensive, these flat regions can be easily travelled around by saving multiple trees with TBR, quickly leading up the slope to better trees. Under SPR, the tree landscape (based on exploration of the smaller cases), seems to have a similar shape but the flat regions are much more extensive, so that finding the uphill slope that leads to the MPT requires travelling around large numbers of trees.

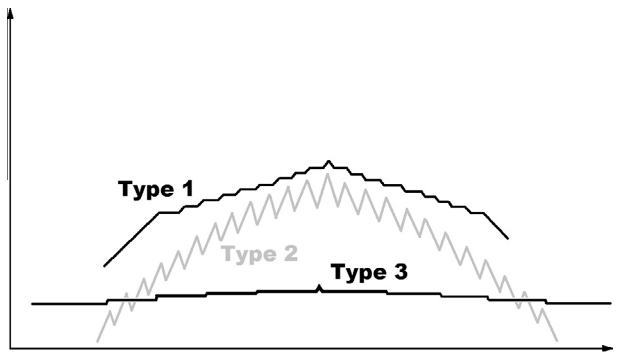


Fig. 1. General depiction of the tree landscape for the three types of data set examined here.

3.3. Behavior of tree search algorithms

The ***xmult*** command of TNT, the only one tested by Radel et al. (2013), is a multi-start method, generally a much more thorough exploration of tree-space than can be done with PAUP* (Swofford, 2001); for each start, constrained and random sectorial searches (Goloboff, 1999) are used, as well as a number of cycles of tree-drifting (for search levels above two). The number of starting points, the cycles of tree-drifting, and the number and size of the sectors to be selected increase with the search level; at the end, the results for each of these starting points are combined with tree-fusing, a genetic algorithm. Thus, the ***xmult*** command used as in Radel et al. (2013) is actually too strong a search effort for these data sets. More properly, the ***xmult*** command misplaces a significant amount of effort, especially for the balanced case, in using multiple starting points, which are subsequently input to tree-fusing. Using a single or very few starting points and subjecting them to drifting or ratchet is enough to quickly find the MPT; sectorial searches are also effective, albeit to a lesser degree. For the balanced case, however, tree fusing works much more poorly than in other data sets. Apparently, the reason for this poor performance of tree fusing is that the large groups in the MPT have a much smaller probability of being found by a given search; the first split of the MPT (that is, a partition with half the taxa on each side) has a probability to be found, in the final tree of a given starting point, of less than 1 in 1000. The probability that two independently found trees to be fused will share that partition is very small, and thus large groups will never be exchanged during fusing. Therefore, the success of the ***xmult*** command depends more on the success of the individual builds (RAS, TBR, sectorial search, drifting or ratchet), which provide input trees for fusing, than on the tree fusing at the end. In the pectinate case, each of the groups in the MPT has a very high probability (between 0.96 and close to 1.00, for $\mathbf{t} = 256$) of being found in a given TBR search, regardless of its size (apparently the groups with the lowest probability are those defined by change in one character reversed a few nodes down or up the tree). Thus, tree fusing can work very effectively for the pectinate case.

Given this situation, the fact that the ***xmult*** command does not save multiple trees with TBR and that it increases the number of starting points with the search level, is in fact counterproductive for data sets of Type 1, especially in the balanced case: saving enough multiple trees under TBR branch-swapping is a simpler strategy that invariably leads to finding the MPT. Tree-drifting (Goloboff, 1999) and/or ratchet (Nixon, 1999) with TNT can be used to mimic the saving of multiple equally parsimonious trees but much more efficiently. As implemented in TNT, both ratchet and drifting alternate cycles where suboptimal trees are accepted with cycles where only trees of equal scores are accepted (in the case of ratchet, this is done with unaltered character weights; this differs from Nixon's, 1999 original method, but is usually more efficient). Accepting rearrangements of equal score means that the tree resulting from such a cycle of swapping will differ from the original tree in several TBR rearrangements. This is in contrast to what happens when saving multiple trees with RAS plus TBR as the only search techniques; each of the trees saved when swapping the first tree will differ from the first by a single TBR move; each of the trees saved from each of those in turn will differ by only two TBR moves; therefore, only after a very large number of trees have been swapped will TBR be able to swap on trees differing from the original by several TBR moves. In the case of drifting and ratchet as implemented in TNT, after a given number of rearrangements have been accepted, the cycle is interrupted, and normal TBR is resumed – now on a tree differing by several TBR moves, but obtained much more quickly than if saving and swapping multiple equally parsimonious trees. Therefore, tree-drifting and ratchet produce almost immediately the MPT for data sets of Type 1.

Table 1 shows the numbers of TBR rearrangements on the MPT for both the balanced and pectinate cases, as well as the number of rearrangements that lead to trees within one step of the MPT. A search that eventually lands on any one of these trees within one step, will quickly find the MPT – it is separated by a single TBR rearrangement. This may explain why TBR is much more likely to find the MPT by swapping from a given starting point in the case of pectinate trees, than in the case of balanced trees: the number of 1-step suboptimal trees that can easily lead to the MPT is much larger in that case.

3.4. Time differences between balanced and pectinate trees

One of the findings of Radel et al. (2013) was that calculating the MPT took significantly longer for pectinate trees. This seems to be in contrast to the fact that, when a single tree is saved, the probability of finding the MPT in a given replication is higher when the trees are pectinate (as discussed under Sections 3.1 and 3.3). But Radel et al. (2013) found that completing the execution of ***xmult*** (at search level 3) for pectinate trees of 8192 taxa took ca. 7.5 times longer than for balanced trees. They considered this as "unexpected", and conjectured that it "may be a property of TNT's search heuristic". However, it is well known (e.g. Allen and Steel, 2001; Felsenstein, 2004) that the number of TBR rearrangements for a tree of $\mathbf{t}$ taxa, while in the general order of $\mathbf{t}^3$, changes with tree topology, and pectinate trees require more rearrangements than balanced trees. For 8192 taxa, the number of rearrangements needed to complete TBR on a pectinate tree (see Table 1) is 3.662×10^{11}, while for balanced trees the number of rearrangements is only 2.617×10^{9}. Thus, the ratio of the number of rearrangements is about 140 to one. This alone suffices to explain that in the case of pectinate trees running the ***xmult*** command with the same parameters takes much more time than for balanced trees, even if an individual starting point has a higher probability of eventually leading to the MPT via branch-swapping.

3.5. Distribution of homoplasy

Another peculiarity of these data sets, not stressed by Radel et al. (2013), is that the homoplasy for the characters, while abundant on average, is not uniformly distributed. If the data are generated following Radel et al.'s Fig. 2, for a balanced tree, then the first five characters in the matrix have no homoplasy in the MPT; then every successive set of four characters has twice the amount of homoplasy as the previous set plus one step. These categories follow the depth of the groups in the MPT (when rooted symmetrically), as shown in Fig. 2A for $\mathbf{t} = 32$. Each tree branch has a single synapomorphy, and all the synapomorphies in the tree are

Table 1
Type 1 data sets, number of TBR moves to the MPT (Neig.size), number of TBR moves within one step of the MPT (1-step), and proportion of TBR moves to the MPT that produce a tree within one step, for the balanced and pectinate cases, and different numbers of taxa **t**.

t	Balanced			Pectinate		
	Neig.size	1-Step	Proportion	Neig.size	1-Step	Proportion
8	149	40	0.26845638	173	62	0.35838150
16	1221	104	0.08517609	1901	366	0.19253025
32	8101	232	0.02863844	18,157	1948	0.10728645
64	46,949	488	0.01039426	159,213	9316	0.05851281
128	249,573	1000	0.00400684	1.334×10^6	41,490	0.03109605
256	1.252×10^6	2024	0.00161553	1.093×10^7	176,762	0.01617802
512	6.044×10^6	4072	0.00067365	8.844×10^7	733,352	0.00829239
1024	2.834×10^7	8168	0.00028819	7.116×10^8	2,995,344	0.00420903
2048	1.301×10^8	16,360	0.00012576	5.710×10^9	12,123,838	0.00212331
4096	5.873×10^8	32,744	0.00005575	4.574×10^{10}	48,817,318	0.00106714
8192	2.617×10^9	65,512	0.00002503	3.662×10^{11}	159,987,156	0.00043684

changes $0 \rightarrow 1$ (i.e. there are no reversals in the MPT). Interestingly, the groups supported by characters with the least homoplasy are the ones that a given individual TBR search will most likely miss (see Section 3.3), and vice versa. In the case of balanced trees, there are also distinct classes of characters, in terms of their homoplasy; on the optimal tree, each of the branches has a single synapomorphy, but many of the synapomorphies are changes $1 \rightarrow 0$ (reversals), and some of the characters change back and forth several times (e.g. in Fig. 2B, for **t** = 25, character 2 has eight changes, half of which are reversals).

3.6. Wagner or random trees as starting point for TBR

Perhaps the most unexpected property of these data sets is in the difference in results between using a random tree or a RAS tree as the starting point of branch-swapping (without saving multiple trees): the probability of finding the MPT is much higher when a random tree is used. Wagner trees (or equivalent forms of “greedy trees”) are almost universally used as the starting points for tree search methods, because they have much better scores than random trees, and then the subsequent rearrangement phase can be completed much faster. However, in some rare cases, Wagner trees may work as a sort of “flytrap”, increasing the chances of the TBR search to start from a tree that either is, or leads to, a local optimum. This can happen even in small data sets and when the Wagner and TBR algorithms are fully randomized. An example is the matrix of Fig. 3: a Wagner tree where every addition and insertion sequence are equiprobable, followed by TBR swapping (on a single tree, equiprobably choosing among all TBR neighbors that improve score), produces the MPT (of 17 steps) with probability 0.6250, and a suboptimal tree (of 18 steps) with probability 0.3750. But if a random tree is used as the starting point (so that each of the 15 possible trees is equiprobably the starting point for TBR), the probability of finding the MPT is now 0.7500, and the same suboptimal tree as before can be found, but now with probability 0.2500. This difference is due to the fact that the Wagner algorithm itself has a probability of 0.3750 of producing the locally optimal tree of 18 steps; each of the three other possible outcomes of the Wagner algorithm are either the MPT (with $P = 0.2500$), or suboptimal trees that unambiguosly lead to the MPT via TBR [(a (b (d (c e)))) with $P = 0.06250$ or (a (e (d (b c)))) with $P = 0.3125$]. The possibility that using a Wagner tree instead of a random tree as starting point for TBR may make the search *less* likely to find the optimal tree has not, to my knowledge, ever been raised before. The calculations for the matrix of Fig. 3 can be done by exhaustive enumeration of all possible outcomes of the Wagner and TBR algorithms, but

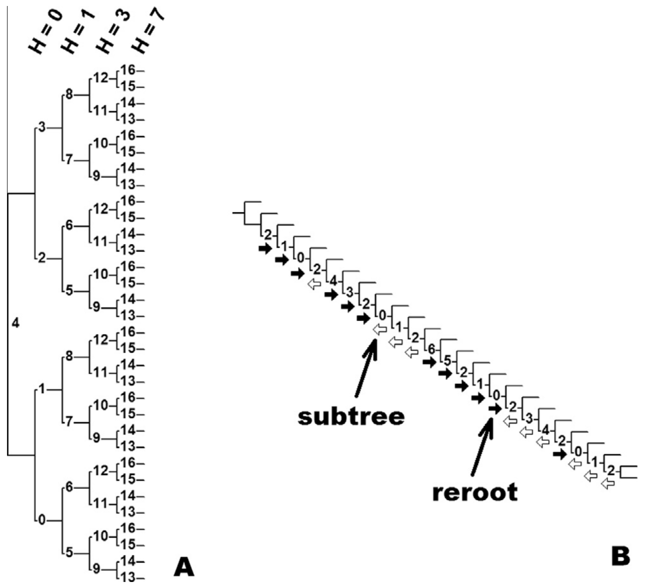


Fig. 2. Synapomorphies for each clade, for data sets of Type 1, in the balanced (A) and pectinate cases (B). The characters are numbered as the data set is generated following the rules of Chai and Housworth (2011) and Radel et al. (2013). Note that for the balanced case, there are no reversals, and the number of parallelisms increases with the depth of the clades (e.g. character 13 changes independently in 8 terminal taxa, so that the homoplasy is 7 steps, and character 5 changes independently in 2 clades, so that there is 1 step of homoplasy). For the pectinate case, the multiple changes in the characters are changes back and forth between the states 0 and 1 (indicated with black and white arrows), nested within each other.

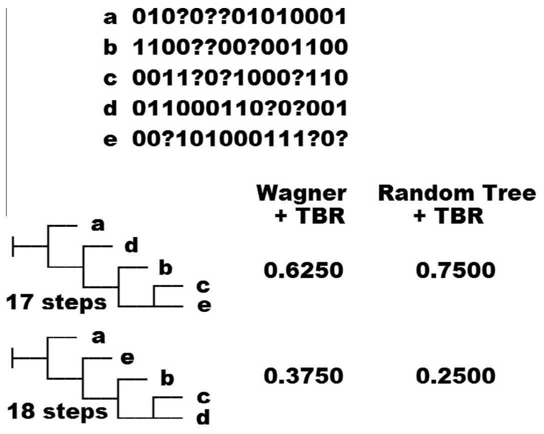


Fig. 3. A case where starting TBR branch-swapping from Wagner trees makes it less probable to find the MPT than starting from a random tree (when the addition, clipping, and insertion sequences are fully randomized). For a Wagner tree as starting point, the probability that the search will find the MPT is $P_{(MPT)} = 0.625$, and the probability that it will find the suboptimal tree is $P_{(sub)} = 0.375$. For the random tree as starting point, $P_{(MPT)} = 0.75$, and $P_{(sub)} = 0.25$. See text for discussion.

such exhaustive enumeration is not tractable for data sets with more than 10 taxa. Actual implementations, like TNT or PAUP*, usually make a number of compromises for the sake of efficiency, so that they may have some determinism (e.g. in using addition or insertion sequences that are not fully randomized). Therefore, in the case of these Chai–Housworth data sets, it is unclear whether the difference in using Wagner and random trees as starting points is due to a deterministic behavior in the search algorithms of the specific programs used here, or would also occur for ideally randomized algorithms. The similarity in results between PAUP* and TNT suggest it is *not* a problem of implementation, since it seems unlikely that the two programs would share an identical bias. Fig. 4 shows the distribution of tree-lengths, for 10,000 replications of each of these two types of starting points for the search, for **t** = 256, when using TNT. The balanced tree is shown in Fig. 4A; in this case, the MPT is found 35 times more frequently when starting from a random tree than when using a RAS (885 hits instead of only 25). In the case of pectinate trees (Fig. 4B), the difference is less drammatic, about two times higher for random trees as starting point (in the case of PAUP*, the frequency of the MPT for the pectinate case with Wagner trees as the starting points is significantly higher than for TNT, similar to that for random trees; data not shown). The taxon sequence in the matrices used for Fig. 4 is randomized, but if the taxon sequence that results from Chai and Housworth's rules for generating the data is retained, the same results are obtained. The taxon sequence might have an effect because TNT does not (by default) randomize the sequence of TBR swapping – it clips and moves the terminal taxa in the same sequence as they are in the matrix. Apparently the same is true of PAUP*, because using a given random tree as the starting point for branch-swapping without eliminating zero-length branches invariably produces the same result, regardless of the current value of the random seed (retaining zero-length

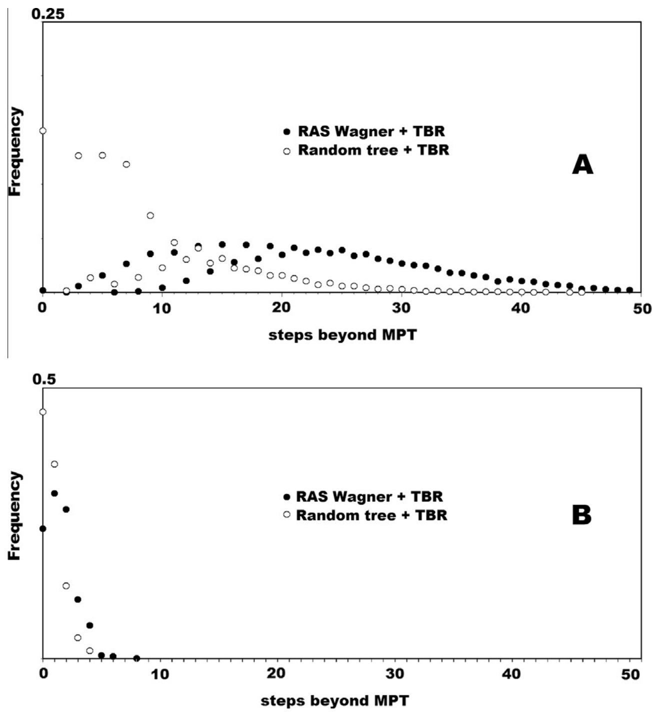


Fig. 4. Frequency of tree lengths found by TNT searches with TBR (single tree = nomulpars) starting from either a RAS Wagner tree or a random tree for data sets of Type 1 (**t** = 256), in the balanced (A) and pectinate (B) cases. Timings in sec. (A) run 10,000 times in Linux; average time: RAS + TBR, 0.0177; random tree + TBR, 0.0254. (B) run 5000 times in Windows; average time: RAS + TBR, 0.0122; random tree + TBR 0.0202.

branches is necessary for this test, because PAUP* randomly resolves polytomies before swapping a tree, thus the resolution chosen would depend on the random seed). For the balanced case, using random or RAS trees in PAUP* produces the same difference observed for TNT – including the difference, near the minimum length of balanced trees, between odd and even numbers of steps beyond the minimum (see Fig. 4A). In TNT, when using random trees as starting point, the probability of finding trees of minimum length for **t** = 256 is 0.1770, while the probability of finding trees with an odd number of steps beyond the MPT (between 3 and 9) is 0.4423, and the probability of finding trees with an even number of steps beyond the MPT (between 2 and 8) is 0.02478 (or 17:1). This difference between the probability of finding trees with an even or an odd number of steps beyond the MPT is also observed when using Wagner trees as starting points.

3.7. Trees within one step of the MPT

In the case of balanced trees, not a single one of the 10,000 searches starting from a random tree, or the 10,000 searches starting from a Wagner tree (Fig. 4A), found trees 1 step beyond the minimum; the final result in all cases was either optimal, or 2 or more steps beyond minimum. This suggests that all the possible trees 1 step longer than the MPT are within one TBR move of the MPT. This idea is very strongly reinforced by swapping from the MPT and accepting rearrangements one step longer: the only trees found are those produced by swapping on the first tree (the MPT itself); swapping on those other trees produces no new trees within one step of the MPT (this was verified with TNT for the cases from **t** = 8 to **t** = 4096). Note that Table 1 shows the number of TBR *moves* within one step, but many of those moves are duplicates: they are different ways of producing the same tree. The number of actually distinct binary trees within one step of the MPT for the balanced case is always 2**t** – 6. This is the number of distinct trees produced by NNI moves – that is, all the trees produced by a NNI move to the MPT, and only those trees, are one step longer than the MPT. This could also be predicted from the distribution of synapomorphies shown in Fig. 2: since every branch has a single synapomorphy, collapsing it and resolving it differently (which amounts to an NNI operation) will increase tree length by one step, but moving any branch to a location more than a single node away will cross boundaries of other synapomorphies (thus adding more steps).

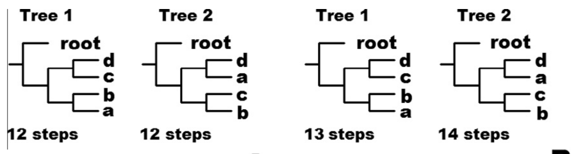
In the case of pectinate trees, every NNI move to the MPT will also increase tree length by one step, but other moves will do so as well, because of the existence of reversals. An example is in Fig. 2B: when the subtree characterized by a change in one direction (e.g. $1 \rightarrow 0$ for char. 0) is rerooted at the first branch below with a change in the opposite direction for the same character (a single TBR move, equivalent to several NNI or SPR moves), tree length is increased by only one step.

4. Low amounts of homoplasy, rugged landscape: data sets of Type 2

Data sets for which local search methods have a low probability of finding the MPT can be designed by modularly combining local optima in different regions of the tree. For four taxa, both SPR and TBR local searches amount to exhaustive solutions (there are only three possible trees in that case, and SPR and TBR will produce the other two trees as possible rearrangements from any of the three). Therefore, the minimum number of taxa for which tree islands could occur is five (four taxa plus root). Consider the two trees shown in Fig. 5A. Moving from one tree to the other requires two TBR moves. A data set for which these two trees are equally parsimonious can be created by assigning two apomorphies to each of the two groups. Then, each of the two trees has the same length ($L = 12$). Branch-swapping starting from one of the trees cannot find the other; if the RAS Wagner tree is fully randomized, then a build has probability 0.5 of finding each of the two trees. For the matrix in Fig. 5A, swapping from every possible starting tree will lead to one of the two optimal trees.

The matrix that produces the two equally parsimonious trees in Fig. 5A can be modified so that one of the trees has a better score than the other, as in Fig. 5B, by adding a single apomorphy for group (ab). Now tree 1 is 13 steps long, one step shorter than tree 2. Branch-swapping on tree 2 cannot find tree 1 – intermediate trees have at least 15 steps. A Wagner tree in which the taxa are added with the "as is" sequence will find the most parsimonious tree, but other addition sequences will not. A Wagner tree in which every addition sequence is equiprobable, and where ties are broken equiprobably, has a probability of 0.75 of producing tree 1, and a probability of 0.25 of producing tree 2; no other tree is a possible outcome of a RAS Wagner tree. Note that even when multiple trees are saved, tree 2 will not produce tree 1 by branch-swapping (local optima and "islands" are identical in this case). Using random trees as starting points for TBR, the probability of finding tree 1 or tree 2 is exactly the same, 0.5.

The pattern in Fig. 5B can be expanded, so that each of the previous matrices is represented by four copies, and additional apomorphies keeping together the four parts (related to each other as in the original matrix) are included. The rest of the matrix is filled with zeros. This is shown in Fig. 6. The pattern can continue being expanded in the same way to any level N (with the matrix of Fig. 5B at level $N = 1$), every time including four copies of the matrix corresponding to the previous level, and nine additional characters at the end determining relationships for the four parts as in the matrix for level $N = 1$. Since this is a very modular way to generate the matrices, the results of applying RAS plus TBR are easily tractable. This could be verified by considering a matrix with the first nine taxa of Fig. 6 – that is, a matrix with two modules, small enough that the exact probabilities of every possible outcome can be calculated by enumeration of all possible addition, clipping, and insertion sequences. The probability of finding the MPT or the three possible local optima for that matrix using Wagner + TBR is exactly the product of the probabilities of the two modules. That is, $0.75 \times 0.75 = 0.5625$ for the MPT, $0.75 \times 0.25 = 0.1875$ for each of the two trees 1-step longer where one of the modules is in the suboptimal configuration, and $0.25 \times 0.25 = 0.0625$ for the 2-steps longer tree where both modules are in the suboptimal configuration (the same is not true for random trees as starting point of branch-swapping, as explained below in Section 4.3, so that the probability of obtaining a specific result in that case cannot be easily predicted). Another experiment doubling all but the first taxon in the matrix of Fig. 5B and adding an apomorphy to hold together each pair of doubled taxa also indicates that the probabilities of resolving relationships between the pairs are identical to those for the original matrix; the trees ((aa′)(bb′))((cc′)(dd′)) and ((aa′)(dd′))((bb′)(cc′)) are then found by a RAS + TBR search with exactly the same probabilities as before, 0.75 and 0.25 respectively (and for this case as well, using random trees as starting point for branch-swapping produces different probabilities; see under Section 4.3).



A

root	00	00	00	00
a	11	00	11	00
b	11	00	00	11
c	00	11	00	11
d	00	11	11	00

B

root	000	00	00	00
a	111	00	11	00
b	111	00	00	11
c	000	11	00	11
d	000	11	11	00

Fig. 5. Construction of data sets with multiple islands. (A) The two trees, 1 and 2, are separated by two TBR moves, and therefore the matrix that has the same number of characters supporting each of the groups has those two trees in separate islands; a TBR search starting from a single point can only find one or the other tree, never both. (B) One of the two trees is now made shorter than the other, because one of the groups is supported by three characters instead of two; the trees intermediate between 1 and 2 have 15 steps or more, and TBR searches can either find the MPT (13 steps, with $P = 0.75$) or the suboptimal tree (14 steps, with $P = 0.25$). See text for additional discussion.

The properties for RAS + TBR searches that can be predicted by the modularity in these matrices are shown in Table 2. The probability of finding the MPT for the matrix of level 1 shown in Fig. 5B is $P_{(1)} = 0.75$. In contrast, finding the MPT for Fig. 6 requires that the proper configuration is achieved for each of the groups shown with gray dots in the tree of Fig. 6 (0.75^4); it also requires that those four groups are in turn related as ((AB)(CD)), since the tree that displays those 4-taxon groups as ((AD)(BC)) is a local (sub)optimum as well; this adds a factor of 0.75. Then (when fully randomizing addition sequences and tie-breaking in Wagner trees), the probability P of finding the MPT for a matrix of level 2 (as in Fig. 6) is $P_2 = 0.75^5 = 0.2373$, and for level N, $P_N = \mathrm{P}_{N-1}^4 \times 0.75$.

The length L of the MPT for these matrices will always be a multiple of 13 steps; for level N, $L_N = 4 \times L_{N-1} + 13$. The number of taxa is $\mathbf{t}_N = 1 + 4^N$. On the MPT, 5 out of every 9 characters are homoplasy free, and 4 have 2 steps; thus, the ensemble consistency index (Kluge and Farris, 1969) is CI = 9/13 = 0.6923. Although the homoplasy is relatively low, as the level increases, the probability of finding the MPT by a single Wagner tree followed by TBR branch-swapping (even if saving multiple trees) decreases quickly; for $\mathbf{t} = 65$ the theoretical probability is $P = 0.002378$ (approximately the frequency observed empirically in TNT with the ***rseed[*** option, which breaks ties in Wagner trees more or less randomly; see below, under Section 4.2). For $\mathbf{t} = 256$ or above, the theoretical probabilities of finding the MPT with a RAS plus TBR are too small to be verified empirically.

The modularity also makes it possible to calculate the number of islands of trees. For Fig. 5B there are two islands of trees. For Fig. 6, each of the groups A, B, C, or D can be resolved in two configurations (2^4), and there are in turn two ways to relate those four groups, so that the number I of islands (each containing a single tree) for level 2 is $I = 2^5 = 32$; one of those trees corresponds to the MPT. This means that a TBR search starting from a single point will necessarily lead to one of these 32 trees; each of the 31 suboptimal trees is formally an "island", because the TBR algorithm cannot escape from that tree to shorter trees, or to trees in the other islands. Note that this is the number of islands *sensu* Maddison (1991); if multiple equally parsimonious trees are not saved (thus finding locally optimal trees, not necessarily "islands"), the number of configurations that can be found doubles, because of small occassional plateaus in the tree landscape. As the level N increases, the number of islands increases rapidly, as $I_N = 2 \times \mathrm{I}_{N-1}^4$ (see resulting values in Table 2). Note that even when each island contains a single tree, the probability of finding the MPT is not simply the inverse of the number of islands, because some islands can be found with higher probability than others when starting from a Wagner tree (e.g. for $N = 1$, the island of $L = 13$ can be found with $P = 0.75$, and the island of $L = 14$, with $P = 0.25$).

The worst possible lengths at which a TBR-based search will arrive can also be calculated. Each of the individual sectors of the tree (e.g. A, B, C, and D in Fig. 6), and the way in which the sectors themselves are related, can independently be in an optimal or suboptimal configuration. Recall that for $N = 1$ ($\mathbf{t} = 5$), optimal length is

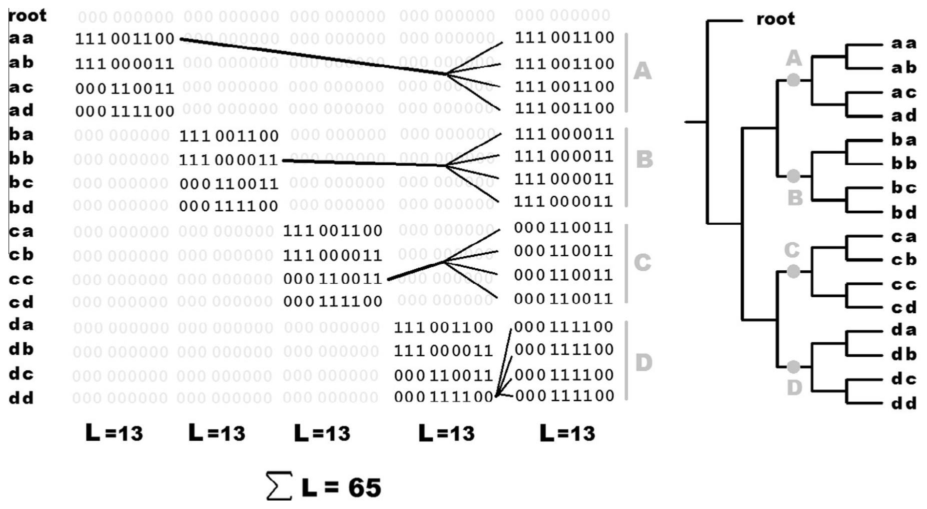


Fig. 6. Expansion of the pattern shown in Fig. 5B. Each module is repeated 4 times, and characters supporting similar relationships for the 4 modules are added. See text for discussion and details.

Table 2
Type 2 data sets. NC, number of characters; $L_{(MPT)}$, length of the MPT; $P_{(MPT)}$, theoretical probability that a given search with RAS + TBR will find the MPT; $W_{(TBR)}$, length of the worst tree that can be found by TBR; $P_{(W)}$, theoretical probability that a given search with RAS + TBR will find a tree of length $W_{(TBR)}$; *G*, largest possible number of steps for the matrix (=length of the bush). In all cases, values are for TBR searches saving multiple equally parsimonious trees. See text for details.

t	NC	$L_{(MPT)}$	$P_{(MPT)}$	#Islands	$W_{(TBR)}$	$P_{(w)}$	*G*
5	9	13	0.7500	2	14	0.2500	18
17	45	65	0.2373	32	70	9.766×10^{-4}	144
65	189	273	2.378×10^{-3}	2,097,152	294	2.274×10^{-13}	864
257	765	1105	2.400×10^{-11}	3.869×10^{25}	1190	6.638×10^{-52}	4608
1025	3069	4433	2.488×10^{-43}	4.479×10^{102}	4774	4.854×10^{-210}	23,040
4097	12,285	17,745	2.875×10^{-173}	8.053×10^{410}	19,110	1.388×10^{-842}	110,592
16,385	49,149	70,993	5.124×10^{-693}	8.411×10^{1643}	76,454	9.279×10^{-3367}	516,096

13, and TBR could get trapped at trees of 14 steps. Thus, the worst case will be when all sectors are in a suboptimal configuration (with 14 steps for the relevant characters), so that the length W of the (single) worst tree that can be produced by TBR is $W_N = 4 \times W_{N-1} + 14$ (see Table 2 for actual values), and the ratio L_N/W_N is a constant 13/14 = 0.9286. In other words, for these data sets a TBR search saving multiple trees is guaranteed to find a tree of W_N steps or less (as already mentioned, a few additional configurations may be locally optimal under TBR if multiple trees are not saved, but this adds only a few steps to W_N). The probability of a given RAS + TBR search to actually reach this worst length as final point is much less than the probability of finding the MPT; for $N = 1$, $P_W = 0.25$, and for $N > 1$, $P_{w,N} = P_{w,N-1}^4 \times 0.25$ (values in Table 2). Evidently, for large **t** values, both the MPT and the worst TBR tree are highly improbable; most of the searches will end at trees with intermediate lengths. Note that the length W_N is the worst length that can be found by searching with TBR saving multiple trees, which is still far from the maximum possible number of steps on any tree (the *G* value; Farris 1989). For $N = 1$, $G = 18$, and for $N > 1$, $G_N = (4 \times G_{N-1}) + (9 \times (t_N - 1)/2)$; for **t** = 4097, the worst possible tree is 5.7871 times longer than the worst tree that can be found by TBR.

4.1. Tree landscape

A representation of the tree landscape for these data sets is shown in Fig. 1 (middle curve). Unlike the case of the Chai–Housworth data sets, where flat regions always delimit with up-hill regions, there are (almost) no flat zones for these data sets. Thus, the landscape is very rugged, composed almost exclusively of isolated peaks and valleys – each of these contains a single or very few trees. This is the sort of landscape that is most difficult to travel with the standard multi-start method of RAS plus TBR, or saving multiple equally parsimonious trees.

4.2. Behavior of tree search algorithms

The empirical observations that can be done with TNT (shown in Table 3) or PAUP* are in line with the theoretical expectations for these data sets. Doing multiple RAS followed by TBR produces the MPT with a frequency that quickly decreases with the number of taxa; for 5, 17, and 65 taxa the empirical results agree rather well with theoretical expectations (with empirical frequencies within 20% of the expected probabilities). For 257 or more taxa, RAS plus TBR never found the optimal trees. These data sets are much more difficult to analyze by standard search methods than the Chai–Housworth data sets. Therefore, for 257 or more taxa, finding the MPT by the standard technique of multiple RAS plus TBR is in practice impossible.

Although RAS plus TBR cannot find the optimal trees except for the smallest cases, saving suboptimal trees can. The option of saving suboptimal trees is rarely used in phylogenetic analyses, because it typically requires saving (and swapping) too many trees to be feasible within short times. Saving and swapping with TNT trees within two steps of the best trees found so far produces the optimal tree for these data sets; if enough suboptimal trees are saved (for **t** = 257, saving up to 150 trees) the MPT is found in the vast majority of cases (97.5% of 1000, using a single RAS Wagner tree as starting point for TBR). This, however, requires swapping on a number of extra trees, a number which increases quickly with the taxa – this strategy takes an average time of 6.15 s for **t** = 257, but is no longer practical for the case **t** = 1025

Table 3
Type 2 data sets, empirical results with TNT. $F_{(R)}$, frequency of the MPT with a random tree as starting point for TBR, taxon sequence unmodified; $^*F_{(R)}$, frequency of the MPT with a random tree as starting point for TBR, taxon sequence randomized in matrix; $F_{(W)}$, frequency of the MPT with a RAS Wagner trees as starting point for TBR; Neig.size, number of TBR moves to the MPT; 1-step and 2-step, number of TBR moves that produce trees within 1 or 2 steps of the MPT. The proportion of moves within 2 steps (2-steps/Neig.size) is shown in the last column. For 5 to 257 taxa, frequency values calculated with 10,000 replications; for 1025 taxa or more, frequency values calculated with 1000 replications; in all cases, TBR saved a single tree.

t	$F_{(R)}$	$^*F_{(R)}$	$F_{(W)}$	Neig.size	1-Step	2-Steps	Proportion
5	0.3315	0.6055	0.7393	31	0	8	0.25806452
17	0.3875	0.2912	0.2444	1559	0	88	0.05644644
65	0.1113	0.0102	0.0029	51,831	0	408	0.00787174
257	0.0013	0	0	1.323×10^6	0	1688	0.00127540
1025	0	0	0	2.942×10^7	0	6808	0.00023141
4097	0	0	0	6.043×10^8	0	27,288	0.00004516

or more. One would think that saving trees within one step instead of two would suffice for the optimal tree to be found. Consider the case of **t** = 5 (Fig. 5B): the intermediate trees between the suboptimal tree and the MPT are two steps longer than the MPT but only one step longer than the suboptimal tree. Yet starting to swap from the suboptimal tree and saving trees within one step fails to find the MPT. This is because an intermediate tree [e.g. (d (c (b a))), of 15 steps] can be found from the 14-steps tree, but then when swapping begins on that tree, the tree of 14 steps is quickly found again and the 15-steps tree is transformed back into that one, instead of being transformed into the 13-steps MPT. Transforming the 15-steps tree into the 13-steps MPT would require using "steepest descent", which adds another layer of extra computational work to the one required by saving suboptimal trees. The chances of this problem happening are increased by the fact that TNT clips (and reinserts) taxa using the same sequence: if the first tree found when swapping from the 14-steps tree corresponds to an alternative position for the first taxon (**a**), then the first tree that will be found when swapping on the 15-steps tree will *also* correspond to an alternative position of the same taxon. These chances are significantly decreased if the taxa are not always clipped in the same sequence, but instead with a randomized one; this option was added to TNT (the ***randclip*** option of the ***bbreak*** command). In such case, the first taxon to move when starting to swap on the 15-steps tree may be **a**, but it may as well be **d** (which produces the MPT if reinserted as sister of **c**), or **c** (which produces the MPT if reinserted as sister of **d**), or the group **ab** (which produces the MPT if reinserted as sister of **cd**). Saving trees one step longer and randomizing the clipping sequence has the big advantage over saving trees two steps longer and swapping with a fixed sequence that fewer trees need to be saved and swapped. For the matrix with **t** = 257, saving up to 50 trees one step longer than the best found so far, and repeating TBR four times changing the random seed (as before, using a single RAS Wagner tree as starting point for TBR), produces the MPT in 991 of 1000 different starting points, but needs only 0.847 s per replication (i.e. over 7 times faster). Note that as the current tree gets closer to the MPT, there are fewer suboptimal trees within one step (zero for the MPT; see Table 3), and thus fewer trees need to be swapped with the successive cycles of TBR. Being significantly faster, the strategy is also applicable for the case of **t** = 1025, where it often produces the MPT within 2–4 min, but it is not fast enough to be applicable for **t** = 4096. The strategy of saving suboptimal trees and repeating TBR with a different clipping sequence is unlikely to produce better results in most real data sets, but it may prove useful in rare specific cases. Searches under implied weights (Goloboff 1993, 2013; Goloboff et al., 2008a,b) are probably one such case, since the tree landscape under that optimality criterion tends to approach the landscape for these data sets – numerous peaks with one or a few trees, separated by narrow valleys, which cannot be easily travelled by just saving suboptimal trees and swapping with the same taxon sequence.

More general search strategies that work well for data sets of Type 2 are the ratchet, tree-drifting, random sectorial searches (selecting enough sectors of a sufficiently large size), and tree fusing. The ratchet and drifting work better if some of the default options are changed; for both ratchet and drifting, the perturbation phase does **r** changes to the tree, and by default **r = t**/8 (but **r** $\leqslant$ 200). Increasing **r** by a factor of 5 (with ***numsub***, and changing the ***giveup*** option, which stops the perturbation phase when a certain percentage of the branches have been clipped and swapped, to ***nogiveup***), both ratchet and drifting can find the MPT with fewer iterations. In the case of drifting, the relative fit difference must also be set to 0.25 (with ***drift: rfitd 0.25 nogiveup numsub 1000***); in the case of the ratchet, increasing the probability of up- or down-weighting characters (e.g. with ***rat: up 10 down 10 nogiveup numsub 1000***) is required. With these changes in parameters, drifting or ratchet can find the MPT by themselves for **t** = 1025 in about 20 s. However, ratchet and drifting are not strictly necessary for these data sets; since (despite the numerous islands) they are highly structured, a combination of sectorial searches and tree fusing works very well. Therefore, the default option of the ***xmult*** command (which uses the results of several replications of RAS + TBR + sectorial search as input for tree fusing; drifting or ratchet to further improve the trees before fusing are only added optionally), finds the MPT very easily.

Additional fine-tuning can be obtained by taking into account that the conflictive solutions correspond to only four tree branches in these data sets, which is unusual. The ***xmult*** command uses the consensus of the final result of the previous replication and the initial stages of the present one, as reference for a constrained sectorial search (Goloboff, 1999) of the sectors that lead (by default) to polytomies of 10 or more nodes (***sec: minfork 10***); lowering ***minfork*** to 3 produces better results for these data sets. Additional tuning that produces faster results for the larger cases is obtained by setting ***mult: wclus 500*** (to speed up the Wagner trees) and ***bbreak: clus 25*** (to speed up branch-swapping); both of these are generally beneficial for any large data set. With this, it is possible to quickly build a sufficient number of starting points and submit them to tree-fusing – producing the MPT in almost every case.

4.3. Wagner or random trees as starting point for TBR

For this kind of data set the results (Fig. 7) of doing TBR (without saving multiple trees, and clipping the taxa in a randomized sequence) when starting from random trees slightly outperform the results of starting from a Wagner tree, in the case of TNT.

As mentioned above, when starting to swap from random trees, the modularity of these data sets is not preserved in the same way as with Wagner trees. This can be verified by considering (as done above for RAS Wagner trees) a matrix with the first 9 taxa of Fig. 6 and calculating the exact probabilities of every possible outcome. The probabilities of the MPT and the three possible suboptimal trees for this 2-module matrix are *not* the product of the

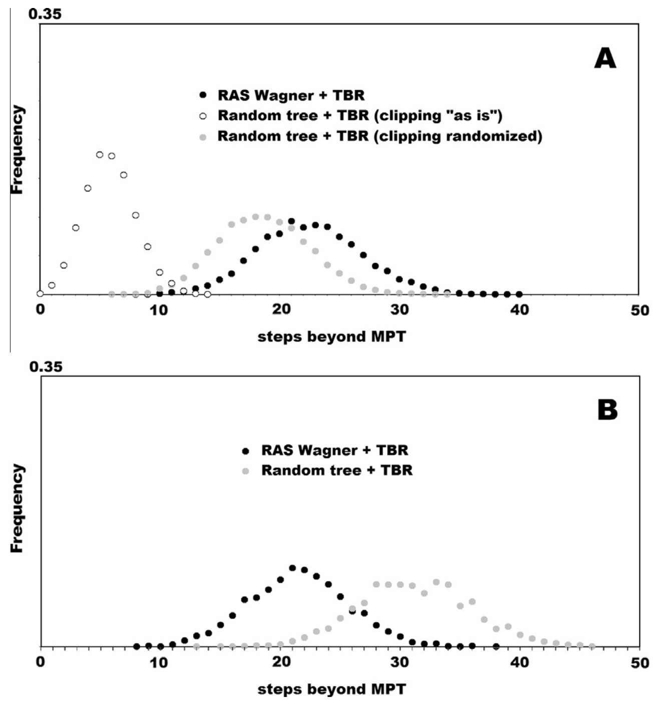


Fig. 7. Frequency of tree lengths found, for data sets of Type 2 (**t** = 257), by TNT (A) and PAUP* (B) searches with TBR (single tree = nomulpars) starting from a RAS Wagner tree, or a random tree. (A) The TNT runs repeated 10,000 times, under Linux; average times (in sec): RAS + TBR, 0.0196; random tree + TBR (sequence "as is"), 0.0392; random tree + TBR (clipping sequence randomized), 0.0456. (B) The PAUP* runs repeated 2500 times, under Windows; average time (in sec): RAS + TBR, 0.1413; random tree + TBR (taxon sequence randomized in matrix), 0.2067.

probabilities of obtaining an optimal (0.5) or suboptimal tree (0.5) for the single module shown in Fig. 5B; the probability of the MPT when starting a fully randomized TBR from a random tree is 0.3242, well above the 0.25 that would be obtained if the modularity were preserved in this context (likewise, the probability of obtaining the suboptimal trees is well below the expected 0.25 for each of the trees; the 2-steps longer tree where both modules are in the suboptimal configuration is found with $P = 0.1877$). A comparable result is obtained when doubling all the taxa but the first in Fig. 5B and adding synapomorphies for each pair of doubled taxa: the tree (((aa′)(bb′))((cc′)(dd′))) has now a higher probability (0.5889) than before (0.5) [evidently, the suboptimal tree (((aa′)(dd′))((bb′)(cc′))) is now found with probability 1–0.5889 = 0.4111].

It may well be that combining more modules to form larger data sets, the same phenomenon continues taking place, and starting TBR from random trees gradually outperforms (in ideally randomized implementations) the use of Wagner trees as starting points. If this is indeed so, the better results obtained with random trees under TNT need not indicate a bias in the branch-swapper (recall that even when the clipping sequence is randomized in TNT, the sequence of reinsertions in the tree is not). While the results for starting from random trees are only slightly superior to the results for starting from Wagner trees when the clipping sequence for TBR (or the sequence of taxa in the matrix) is randomized, the results are dramatically better when the clipping sequence follows that in the MPT – this *is* clearly a case of bias, and also means that there *must be* some specific clipping sequences which produce a poorer result. In the case of PAUP*, the results of starting from random trees are inferior to the results of starting from Wagner trees, and show no significant differences whether the taxon sequence in the matrix is randomized or not. The difference between PAUP* and TNT is only in the results starting from random trees; Wagner trees for PAUP* and TNT have similar length distributions (see Fig. 7; note that the distribution curves for TNT may be less even for the Windows versions, since they use a different random number generator, the one provided by the Open Watcom library). The reasons why swapping from random trees on matrices with a randomized taxon sequence may produce better results in TNT than in PAUP* are not clear; in principle, since both programs do the same series of well-defined rearrangements, the end results might be expected to agree.

5. No homoplasy, extremely flat landscape: data sets of Type 3

The third type of data set studied here is formed by creating a character for each internal branch of the model tree, a character with some missing entries but providing an unambiguous, homoplasy-free synapomorphy for the group **g** defined by the branch. Unlike the two previous types of data set, the model tree can have any (binary) topology or any number **t** of taxa. Each of the characters will have a state 1 in **n** members of the group defined by the left descendant **l** of the corresponding branch, and a state 1 in **n** members of the group **r** defined by the right descendant. It will also have a state 0 in **n** members of **s**, the sister group of **g**, and state 0 for **n** of the remaining taxa that do not belong to either groups **s** or **g**. All the rest of taxa will have a missing entry. As **n** increases, the resulting matrix will converge onto the full matrix representation of the model tree (known as "MRP" after Ragan, 1992), but for low values of **n** the matrix will consist mostly of missing entries. The type of construct resulting when **n** = 1 has been well studied by mathematicians (e.g. Steel, 1992; Böcker et al., 2000; Steel, 2014). Under such conditions, the model tree is guaranteed to have a length $L = \mathbf{t} - 3$, but this length may occur in other trees as well [Steel, 1992 gives as example the quartets ab|cf, bc|de, and ad|ef, for which there are two MPT's, (a (b (c (d (e f))))) and (a (d (e (b (c f)))))]. When **n** = 1, it is possible to determine in polynomial time (Böcker et al., 2000) whether the characters fit a unique tree, and finding the tree (although I am not aware of any actual implementation of this algorithm). For any **n**, the supertree algorithm of Goloboff and Pol (2002, implemented in TNT) can also be applied, finding the model tree (when uniquely defined) very quickly (e.g. <1 s for **t** = 500).

The data sets of Type 3 are those formed when the members to be assigned the 0 and 1 states are selected in the same sequence for all **t** – 3 characters, in which case the model tree is also guaranteed to be the *only* tree of length *L*. To do this, a random ordering $\mathbf{R_t}$ of the taxa, 1, 2...**t**, is defined; then for each of the four sets of non-missing states, the list is scanned separately from the beginning to the end, and the first **n** taxa that belong to the group (or to neither group **g** nor **s**, in the last set of 0's) are selected (when **l**, **r**, or **s** are themselves terminals or small groups, or **g** and **s** are groups with most of the taxa, fewer than **n** taxa may be selected for some of the four sets). In this case, there will be a few terminal taxa that concentrate observed entries, and many taxa with only a few observed 0's or 1's. In the discussion that follows, unless otherwise indicated, **n** = 1.

5.1. Tree landscape

In contrast to the case of data sets of Type 2, the tree landscape for these data sets cannot be predicted from modularity, only observed empirically in the smaller cases (and assumed to be similar in the larger). Thorough searches for smaller cases reveal that these data sets have a single peak but an extremely flat landscape; thus, starting the tree search from any tree and branch-swapping (with either TBR or SPR) saving multiple trees, guarantees finding the MPT. However, numerous individual trees are locally optimal under TBR (i.e. their TBR neighborhood contains many trees of the same score, but no better tree). Therefore, the approach of using one or a few starting points and then saving and swapping multiple trees is possible only for modestly sized data sets, because the number of trees that need to be saved makes the approach prohibitive. For data sets with only 20 taxa, saving and swapping up to 10,000 trees is not enough to consistently find the MPT; even saving 20,000 trees occasionally fails to find the MPT. For 30 taxa, saving up to 50,000 trees finds the MPT only in about a third of the cases. As the number of taxa increases, swapping the trees requires looking at many more rearrangements; for **t** = 50, saving and swapping 50,000 trees takes much longer than for 30 taxa, but 50,000 trees is totally insufficient for this many taxa – searches from a single starting point saving up to 50,000 equally parsimonious trees commonly find trees 4 or 5 steps longer than the MPT. This shows that consistently finding the MPT by saving equally parsimonious trees quickly becomes unfeasible as **t** grows because it requires saving and swapping millions of trees. The resulting tree landscape is shown in Fig. 1 (lower curve). A standard search which tries to find the MPT by swapping multiple trees is like a swimmer stranded in the middle of the sea: it can easily move in any direction, but getting to the shore is next to impossible.

Another indication of the difficulties that can arise with these data sets is in how the number of equally parsimonious trees multiplies enormously as a single character is deactivated, the more so when the character can be mapped as a synapomorphy of an even sized partition (deactivating characters that distinguish groups with two taxa produces the fewest additional trees). When **n** = 1, the maximum number of steps a character can have is 2. Therefore, the trees that previously were suboptimal and become most parsimonious when deactivating a single character must necessarily have (for the full data set) a single step beyond the MPT. For **t** = 100, deactivating the character for one of the groups in the middle of the tree produces hundreds of thousands of equally parsimonious trees, and a completely unresolved consensus. Deactivating taxa has a similar effect; since every taxon in the matrix has at least one non-missing entry (either 0 or 1), deactivating a taxon makes the corresponding character(s) uninformative (now consisting of two 0's and one 1, or one 0 and two 1's), producing the same result as deactivating character(s). Therefore, only the entire matrix can determine a unique MPT; any alteration to the full matrix will create an enormous ambiguity and very large numbers of equally parsimonious trees.

5.2. Behavior of tree search algorithms

The tree landscape of these data sets makes it very difficult for all search methods to find the MPT. Although the MPT for these data sets can be found with specific algorithms that do not use searches (such as Böcker et al.'s, 2000, or Goloboff and Pol's, 2002), in the case of real data sets resembling matrices of Type 3, those algorithms will probably be inapplicable, because some homoplasy and character conflict will exist. Thus, search methods are needed for such real cases, and they need to be improved to be able to find (or at least approximate) the MPT for these data sets.

The reasons why Wagner trees and TBR fail to find the MPT are different. In the case of Wagner trees, the initial stages in the formation of the tree will usually correspond to taxon subsets for which the characters are uninformative. Given that most of the taxa have a very large proportion of missing entries (with observed entries corresponding to different characters in different taxa), most of the subsets of a few taxa will have a single or no 0 states, and/or a single or no 1 states, for the majority of characters. For **t** = 100, the probability of a (random) subset of 10 taxa being completely uninformative is about 0.99. Even if some of the characters are informative at a given stage in the formation of a Wagner tree, they are extremely unlikely to fully determine the subtree that corresponds to the MPT (see Section 5.1, for the effect of deactivating individual taxa on the number of MPT's). Thus, for any medium or large **t**, it is virtually impossible for a Wagner tree to find the MPT, despite all the characters being completely congruent. For **t** = 100, the average length (10,000 replications) of a RAS Wagner tree is 138.94 steps (over 40% longer than the MPT, 97 steps); about 1 every 1000 replications finds a tree which is 25% longer than the MPT, all others do worse. Compared to most real data sets, this is an extraordinarily poor result for Wagner trees (e.g. Wagner trees for the 500-taxon "Zilla" *rbcL* data set of Chase et al. (1993) typically have 2% more steps than the MPT's), specially considering that the longest possible trees are only twice the length of the MPT. Random trees are thus only slightly worse than RAS Wagner trees (about 15% longer, on average; for Zilla, random trees are ~130% longer than Wagner trees).

In the case of branch-swapping, the very flat tree landscape makes it very difficult for TBR to efficiently move along the surface. When a single tree is saved during swapping, the average final lengths are 20–25% longer than the MPT (e.g. for **t** = 500, the best tree out of 2500 replications was 593 steps instead of 497, or 19.3% longer; in the case of Zilla, only 100 replications of RAS + TBR are enough to find trees within 0.055% of the MPT's). Even if multiple trees are saved, things do not improve much, because the number of trees that need to be saved for medium or large sized data sets is too large.

The results obtained with the ***xmult*** command of TNT are better, but still much longer than the MPT. At level = 5, for **t** = 100, an ***xmult*** search uses 9 starting points and incorporates CSS and RSS (consensus and random sectorial searches; see Goloboff, 1999), as well as 4 iterations of tree-drifting (with otherwise default parameters), fusing the 9 trees resulting from each of the starting points with 5 rounds of tree-fusing. The average final length for this routine (repeated 2500 times) is 112.99 steps (16% longer than the MPT), and the best length 108 steps (11.3% longer than the MPT). The final points of different individual searches for these data sets are very different; thus tree-fusing can attempt very few exchanges (producing no significant improvement), and the method of Goloboff and Farris (2001) for estimating the consensus produces a complete bush.

In sum, none of the existing search methods can produce trees even remotely close (within 5% or less) to the MPT for this type of data set. The next two sections describe modifications of the existing building and rearrangement operations that can produce trees that are much closer to optimal.

5.3. Better building methods

The standard method for building Wagner trees is randomizing the sequence with which taxa are added to the growing tree, RAS. Farris (1970) proposed some alternatives, one of which has long been implemented in PAUP* as the "closest" addition sequence (recently implemented in the ***mult*** command of TNT as the "***cas***" option). In PAUP*, the closest addition sequence is unique for each matrix and it is not possible to do multiple replications with this option (unless multiple matrices with the taxa in a different sequence are created); TNT randomly breaks ties in the length increment for several taxa, so this allows doing multiple replications. With the closest addition sequence, each of the taxa not yet placed in the tree are tried for insertion at each branch, and the taxon leading to the smallest length increment is chosen. This requires trying each of the taxa not yet added to the tree, and thus is significantly slower than a RAS Wagner tree. While in rare specific cases the closest addition sequence may outperform the standard RAS (e.g. in randomly generated data sets), it is unusual that it does so. Even more, in the particular case of Type 3 data sets, it is not only slower than the RAS, but also produces significantly inferior results. The tree length distribution obtained for **t** = 100 is shown in Fig. 8A (CAS, gray); compare this to the distribution obtained for RAS (black). The criterion used to select the next taxon to add is actually counterproductive in this case: those taxa for which the characters continue being uninformative tend to produce smaller length increments, and thus will be added first.

Since the reason Wagner trees produce such strongly suboptimal results is that most taxon subsets are uninformative, solving that problem requires that the taxa be added to the tree in such a way that the subsets are maximally informative in the initial stages. Two ways to do this have been recently (November 2013) implemented in TNT, and they produce similar results for these data sets. The first is the "selected" addition sequences or SAS (the "***sas***" option of the ***mult*** command). In this case, the 4-taxon subset for which the three possible trees maximally differ in length is chosen initially (but the outgroup is always part of these 4 taxa; with the "***rseed*** **>** " option, applicable only when all character state transformation costs are symmetrical, the outgroup is chosen randomly for each replication). Subsequent to this, each position for each of the remaining taxa is tried, and the taxon with the maximum difference D between the length for best and worst positions is selected (again, ties in D are broken randomly, so that multiple "selected" addition sequences can be used). This is significantly slower than the RAS, but produces much better results for data sets of Type 3 (data not shown, similar to the IAS described below but slower). Much of the extra work needed for a "selected" addition sequence can be avoided with an "informative" addition sequence or IAS (with the "***ias***" option of the ***mult*** command; note that ties in the taxon selection are also broken randomly, so that different IAS can be tried). This consists of simply counting the number of characters that are informative for different taxon subsets (in the case of nonadditive characters, a character is informative when there are at least two states which occur in at least two taxa); unlike the previous case, the informative addition sequence can be established prior to building the Wagner tree. The results produced by the informative addition sequences for Type 3 data sets when **n** = 1 are spectacular; they are shown in Fig. 8A (white circles). Note that Fig. 8A shows the best length out of 10 addition sequences followed by TBR, which produces better results than a single addition sequence (and decreases the dispersion in the values). In the case of **t** = 100, over 40% of the builds using ten different IAS find the optimal tree. Although in Fig. 8A the builds are followed by TBR (with no mulpars), the difference in results for IAS, RAS, and CAS for building the initial tree is so profound that the differences remain after branch-swapping.

The difference between IAS and RAS, however, is greatly decreased when **n** > 1. There are two reasons for this. First, the RAS produces better results, because there is a smaller probability of a taxon subset being uninformative as **n** grows; the most frequent result for **n** = 1 is about 20 steps longer than the MPT (see Fig. 8A), while the most frequent result for **n** = 3 is about 15 steps longer than the MPT (see Fig. 8B), so that the RAS curves are similar but shifted about 5 steps towards the MPT for **n** = 3. Second, the IAS produces worse results, probably because the difference in informativeness between different taxon subsets is not as marked as when **n** = 1, thus making it more difficult to choose different addition sequences. The results for **t** = 100 and **n** = 3 are shown in Fig. 8B (white circles). Although the difference is much smaller than for **n** = 1, the results for IAS continue being better than the results for RAS, and no cases where IAS actually produces (on average) worse results than RAS have been found. Unfortunately, real data sets are more likely to mimic the situation observed for the case when **n** > 1, so additional methods to continue improving the trees are needed. Note, however, that standard search methods perform extremely well when $\mathbf{n} \gg 1$; the problematic cases are those where **n** is above 1 but still small relative to **t**.

For real data sets, the IAS may produce better results for phylogenomic data sets with uneven taxon representation for the different genes and many missing entries, such as McMahon and Sanderson's (2006) "sparse" 2228-taxon matrix. On this matrix, IAS + TBR produces trees 185 steps shorter than RAS + TBR, on average (200 runs of each of the two routines were made). Even when the Wagner tree is followed by TBR (much more exhaustive), the influence of the initial point still makes a difference in the final result. The IAS requires more computational work than the RAS, to select the initial quartet, and to select successive taxa. That work can be lessened if, instead of making this choice from all possible terminals, the first **i** taxa in a randomized list are examined – that is, by defining a value of maximum allowed "lookahead". For the

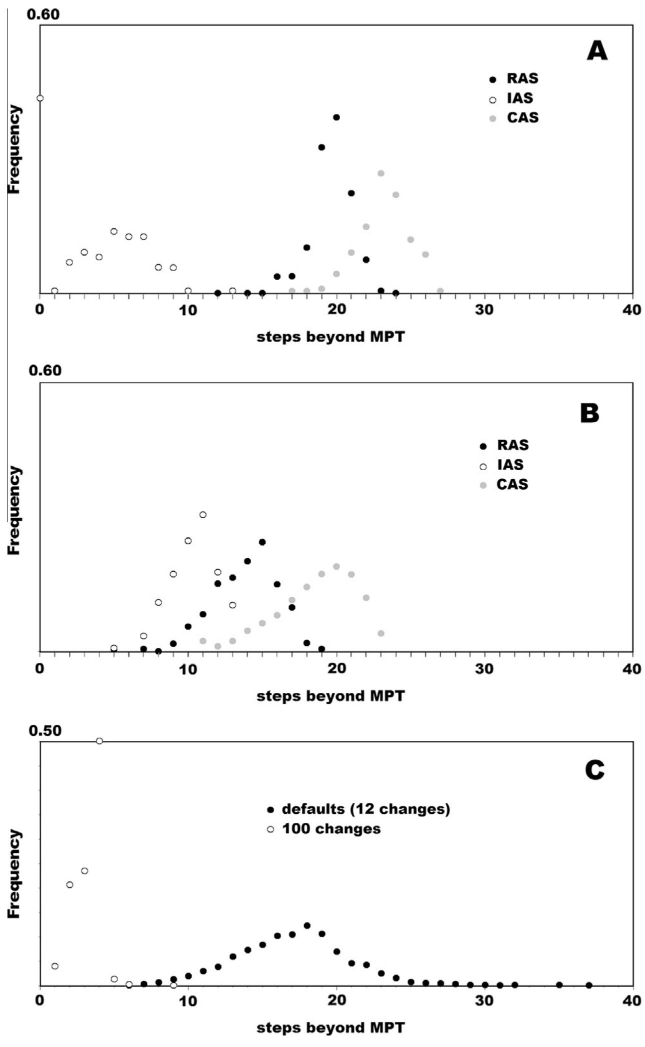


Fig. 8. Frequency of tree lengths found, for data sets of Type 3 (**t** = 100). (A) Comparison of TBR (saving a single tree), when using as starting point 10 different RAS, IAS, or CAS Wagner trees, on a data set with **n** = 1. Each routine repeated 5000 times in Linux. Average run times (sec): RAS + TBR, 0.1814, IAS + TBR, 0.1408; CAS + TBR, 0.3502. (B) Same as A, but for a data set with **n** = 3. Average run times: RAS + TBR, 0.1664; IAS + TBR, 0.0674; CAS + TBR, 0.2782. (C) Comparison between default and modified tree drifting, for data with **n** = 3. The default drifting for **t** = 100 interrupts the perturbation phase when 12 changes have been made to the tree; the modified routine interrupts the perturbation phase only after 100 changes, thus using a larger proportion of time in moving through equally parsimonious tree than in the standard TBR (which only accepts rearrangements that improve score). Each of the two routines (default and 100 changes) was run 1000 times, under Linux, for 40 s (run times for both routines are identical).

comparisons just mentioned in the 2228-taxon matrix, a look-ahead of 150 was used. This decreases the work needed for IAS, and even if it still takes longer than the RAS, it subsequently requires a shorter time for TBR to complete swapping (because the Wagner method delivers a tree which is closer to optimal), and then the total average time with IAS and a lookahead of 150 was only 10% longer than with RAS (~155 vs. ~140 s).

5.4. Better methods for improving existing trees

As mentioned above, the problem these data sets pose to standard search algorithms is that the tree landscape is too flat – saving multiple trees under TBR guarantees finding the optimal tree, but the number of trees that need to be saved is too large for this to be practical. Two minor modifications of the existing algorithms in TNT produce results that strongly outperform the default settings.

The first is letting the perturbation phase in tree-drifting effect to the tree many more changes than the default before switching to normal TBR. In TNT, this is controlled with the ***numsub*** and ***giveup*** options of the ***drift*** command (already discussed for data sets of Type 1, see Section 4.2). This amounts to saving multiple trees, but allows swapping on trees differing by several rearrangements without having to swap completely through thousands of trees differing in one or a few. Fig 8C shows the comparison between tree-drifting with the default parameters (black) and making as many changes to the tree (in every perturbated cycle) as the number of taxa, for the case **t** = 100 and **n** = 3. Each of the two options was automatically interrupted after 40 s (using the ***timeout*** option of TNT), so that they used exactly the same amount of search time. It is clear from Fig. 8C that the additional number of changes makes a significant difference, and produces much better results (i.e. strongly shifted to the left), relative to the default tree-drifting or to the application of IAS + TBR alone.

The second improvement is using sectorial searches to explore a diversity of trees of the same score, instead of (as typically done) using them only to find better trees. This is done by searching trees for the reduced data sets, and accepting all solutions that match the best score (instead of accepting only those that improve it, which is the default option). This can be done by using the ***equal*** option of the ***sectsch*** command (or, in Windows versions, by checking on the "accept rearrangements of equal score" in the Sectorial Search "settings" option of the dialog for the menu option Analyze/NewTechnologySearch). The comparison between these two options is shown in Fig. 9A. Given that for these data sets finding the MPT is difficult even for relatively small cases, the settings for the sectorial search were chosen so that the reduced data sets are smaller than usual (e.g. about 20–30 nodes for **t** = 100, about 50–80 for **t** = 256). That is, an "exclusive" sectorial search (XSS, with non-overlapping sectors that cover all the tree) was used, dividing the tree in 5–3 parts. For **t** = 100 and **n** = 3, this routine alone was sufficient to consistently find the MPT (in about 3/4 of the cases). For **t** = 256 and **n** = 3 (Fig. 9B), this also produced much better results than the default (gray vs. black), but it was insufficient to approach minimum length. In that case, using a IAS Wagner tree for the analysis of reduced data sets produces results approaching minimum length (white circles). Note that when **n** > 1 the IAS option for Wagner trees is not very useful if combined only with TBR for the full tree (as shown in Fig. 8B), but it does improve the results significantly if used to analyze the reduced data sets in combination with XSS.

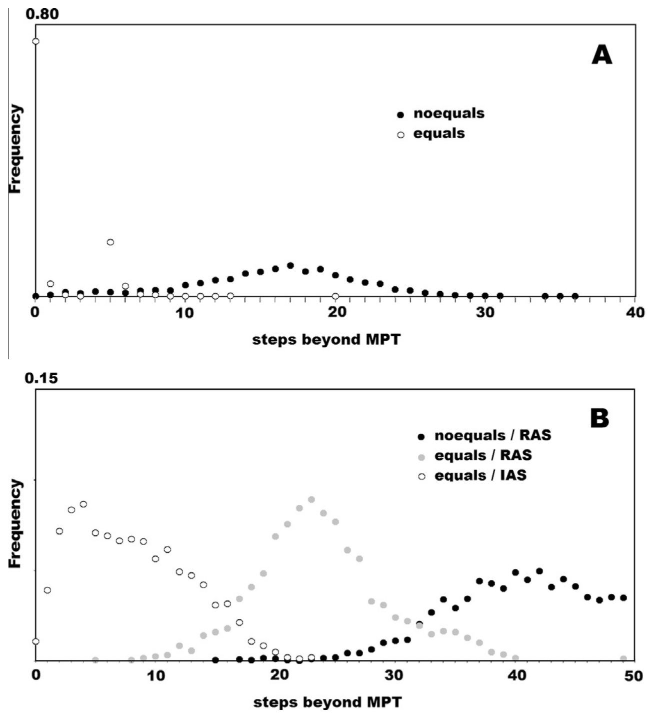


Fig. 9. Frequency of tree lengths found for data sets of Type 3 (**n** = 3), using exclusive sectorial searches (XSS) which either accept only solutions that improve the score for the sectors (the ***noequals*** option, default in TNT), or accept solutions that match the score (the ***equals*** option). All cases run in Linux; average times in sec. (A) Case of **t** = 100, a RAS + TBR as starting point, followed by sectorial search dividing tree in 5–3 parts, 500 times, and using 8 RAS + TBR to analyze each sector (***sectsch = xss5–3 + 500–1 start 8 godri 100***), with ***equals*** (average time 9.7) and ***noequals*** (average time 9.0). Each routine run 2500 times; the default routine (***noequals***) found the MPT a single time, the ***equals*** routine found it 1881 times. (B) Case of **t** = 256, run with the same commands, and also using IAS + TBR to analyze each sector. Average times: ***noequals***, 79; ***equals*** and RAS + TBR 92; ***equals*** and IAS + TBR, 141 s. The best case (out of the 2500 runs) for ***noequals*** was 15 steps longer than the MPT; the best case for ***equals*** and RAS + TBR was 5 steps longer than the MPT (a single case); ***equals*** and IAS + TBR found the MPT in 27 cases (907 cases found trees within 5 steps of the MPT; 730 cases found trees within 4 steps of the MPT – better than the best case for ***equals*** and RAS + TBR).

5.5. Trees within one step of the MPT

The number of trees within one step of the MPT in these data sets (when **n** = 1) is very large, as already discussed above (under Section 5.1). There is however an unexpected detail in the number of TBR moves for data sets of Type 3: when **n** = 1 and the first split in the tree used to generate the data leads to a terminal (i.e. when a terminal is the outgroup), then for a given **t** the number of TBR moves to the MPT that produce trees within one step is a constant number, regardless of the shape of the tree and regardless of the random ordering $\mathbf{R_t}$. This is shown in Fig. 10, for $20 \leqslant \mathbf{t} \leqslant 100$ (gray); for each **t** value, the plot shows the number of TBR moves within one step, for 50 different tree topologies and random orderings $\mathbf{R_t}$. The 50 values are identical for each of the 50 matrices, so they overlap completely. The number of TBR moves to the MPT (in black, with considerable spread), is plotted for the same 50 different tree topologies and orderings $\mathbf{R_t}$ used to calculate the number of TBR moves within one step for each value of **t**.

Experiments confirm that each of those moves producing a tree within one step is also equivalent to an SPR move: the same number of distinct trees within one step are found by swapping from the MPT with SPR or TBR (unlike data sets of Type 1, swapping on those other trees quickly leads to finding additional trees that differ between SPR and TBR; the identity is in the tree sets produced by swapping the MPT alone). However, the number of distinct trees within one step found by SPR or TBR moves to the MPT is not in itself constant: it changes with tree shape and $\mathbf{R_t}$. The constancy in the number of TBR moves is a consequence of the *duplication* of moves in TBR; e.g. given the trees (a(b(c(de))))

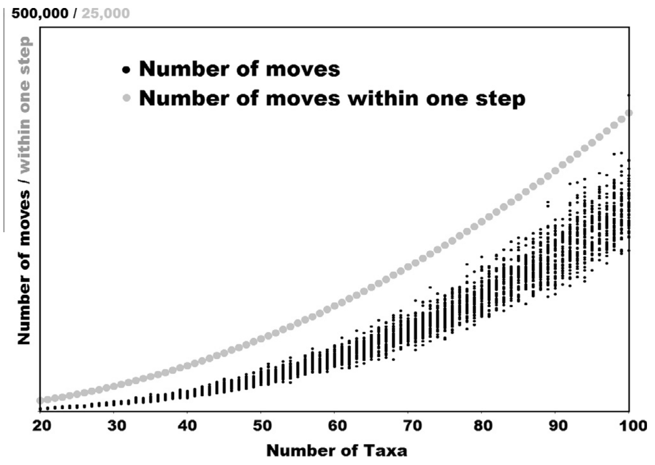


Fig. 10. Number of TBR moves on the MPT (black), and number of TBR moves producing a tree within one step of the MPT (gray), for data sets of Type 3 (**n** = 1), with different numbers of taxa, **t**. For each number of taxa, 50 different model trees and random orderings $\mathbf{R_t}$ were used. The *y*-axis for number of moves, and for number of moves within one step of the MPT have a different scale. There is significant variation in the number of TBR moves, but the number of moves within one step is constant for a given value of **t**. See text for discussion.

and (a(b(e(cd)))), three SPR moves can convert the first into the second, but in addition to those there is a fourth move under TBR (clip **cde**, reroot such that **e** is sister group to **cd**, and reinsert at the original position). Thus, the number of SPR moves to the MPT that produce a tree within one step is not a constant number. This was unexpected, especially because theory predicts that the number of TBR moves to a tree of **t** taxa changes with the tree topology while the number of SPR moves does not (see Semple and Steel, 2003: 33).

When **n** > 1, or when the tree used to generate the data does not have a terminal taxon in the first split, then the number of TBR moves within one step of the MPT for a given value of **t** is no longer constant. Likewise, the number of TBR moves leading to trees within two steps of the model tree is not constant under any circumstances; this is true only for trees within a single step (no doubt, this is connected to the fact that no character can have more than a single extra step when **n** = 1; a tree two steps longer than the MPT necessarily has homoplasy for two characters).

6. Discussion and conclusions

The three types of data sets discussed in this paper have different characteristics, and pose different problems to tree search algorithms. The fact that the MPT can be known exactly for these data sets allows determining the efficacy of different search algorithms, which in turn better illuminates the reasons why tree searches may fail to find the MPT, suggesting ways to correct those problems.

The data sets of Type 1 and 3 have a similar landscape, with only one island containing the MPT, and successively lower plateaus (as shown in Fig. 1, top and bottom curves). Just using TBR saving multiple trees is enough to guarantee finding the MPT, regardless of the starting point used to initiate branch-swapping. This allows finding the optimal trees rather easily for data sets of Type 1, but in data sets of type 3 the plateaus are too extensive for the trees found by TBR to effectively connect to the trees in adjacent, higher plateaus. For these data sets, only saving and swapping hundreds of thousands of trees allows approximating the MPT, but this quickly becomes computationally impossible as the number of taxa increases. Thus, special search strategies are required. The strategies proposed (IAS or "informative addition sequences", accepting trees of equal score for reduced sectors in sectorial searches, and letting the perturbation phase of tree-drifting or ratchet perform many changes to the tree before switching back to normal TBR) allow searches to produce trees of better score, hundreds or thousands of times faster than would be required for more standard strategies. The IAS tries to identify more informative taxon subsets when building the Wagner tree. The strategy based on tree-drifting uses the perturbation phase to quickly produce a tree differing from the initial one in numerous TBR rearrangements (which, under normal TBR saving multiple trees, requires the complete swapping of huge numbers of trees). The strategy based on sectorial searches uses the analysis of the sectors to effectively move around within the present plateau, eventually coming near the edge of higher plateaus. Although the alternative options for tree-drifting and sectorial searches were available in TNT, they are not included as default options, and hence were very rarely used or explored. The present analysis shows that changing those parameters of tree-drifting and sectorial searches improves the results by a huge factor, for data sets where the tree landscape has extensive plateaus (i.e. plateaus with very numerous trees of equal score around a given tree, only one or a few of which can lead to better trees). The comparison between data sets of Type 1 and Type 3 also shows clearly that what makes tree searches difficult is, primarily, the shape of the tree landscape, more than just the existence of homoplasy; data sets of Type 3 lack homoplasy, yet they are the most difficult data sets examined here.

Data sets of Type 2, in contrast, present an extremely rugged landscape, not because of high homoplasy, but instead because of the peculiar distribution of the little homoplasy that is present. For these data sets, the standard search methods using branch-swapping and different starting points are doomed to fail. However, the more elaborate techniques implemented in TNT (sectorial searches, perturbation algorithms, and tree-fusing) produce very good results for these data sets. These data sets also allowed to identify problems that may arise with branch-swapping saving suboptimal trees. Saving suboptimal trees is of no help when searches can easily find numerous equally optimal trees, but it may be useful when the searches frequently land on local optima containing only one or a few trees. These narrow peaks and valleys are likely to occur in analyses under different weighting methods (e.g. implied weighting) or maximum likelihood. Specifying a maximum acceptable excess in the optimality score, and swapping on the resulting trees, requires much larger thresholds if the sequence of clippings in TBR is always the same. Randomizing the clipping sequence allows finding the MPT much more efficiently, specifying acceptance of a smaller excess of steps (thus, fewer trees need to be saved and swapped). The implementation of suboptimal trees in TNT uses this specification of difference in steps beyond the best tree found so far, and the acceptable absolute score is automatically changed as the search finds better trees. This is different from the implementation of suboptimal trees in PAUP*, which requires specification (with the ***keep*** value) of an absolute score for a tree found by swapping to be acceptable. Even if the length of the MPT was known when analyzing a data set like those of Type 2, an implementation like PAUP*'s is not very practical. A search using a RAS is likely to find trees a dozen or more steps beyond the MPT. PAUP* would then require specifying a ***keep*** of trees beyond one step of that value, until a better one is found, then repeating the procedure with a new ***keep*** value; this could require dozens of manually driven searches, with different ***keep*** values, before the MPT is found.

All the findings and new methods presented in this paper were made possible by studying data sets where the MPT is known but hard to find. Although a lot of research has already been invested in tree search techniques, it is clear that further improvements are possible for special cases. The properties of the tree landscape that make finding the MPT difficult for the data sets studied here can also occur in real circumstances, and thus, in addition to their theoretical interest, some of the new methods described here may find practical application in the future.

Acknowledgments

I thank Salvador Arias, Santiago Catalano, Mark Simmons, Mike Steel, and Claudia Szumik for discussion, and for identifying relevant literature. Comments by two anonymous reviewers helped improve the manuscript. Financial support came from CONICET (PIP 112 201101 00687), and a Grant (to J. Cracraft and L. Lohman) entitled "Assembly and evolution of the Amazonian biota and its environment: an integrated approach" (from NSF, NASA, and Fundaçao de Amparo à Pesquisa do Estado de São Paulo).

Appendix A. Supplementary material

Supplementary data associated with this article can be found, in the online version, at http://dx.doi.org/10.1016/j.ympev.2014.06.008.

References

Allen, B.L., Steel, M., 2001. Subtree transfer operations and their induced metrics on evolutionary trees. Ann. Combinatorics 5, 1–13.
Böcker, S., Bryant, D., Dress, A., Steel, M., 2000. Algorithmic aspects of tree amalgamation. J. Algor. 37, 522–537.
Chai, J., Housworth, E.A., 2011. On the number of binary characters needed to recover a phylogeny using maximum parsimony. Bull. Math. Biol. 73, 1398–1411.
Chase, M.W., Soltis, D.E., Olmstead, R.G., Morgan, D., et al., 1993. Phylogenetics of seed plants: an analysis of nucleic sequences from the plastid gene rbcL. Ann. Mol. Bot. Gard. 80, 528–580.
Farris, J.S., 1970. Methods for computing Wagner trees. Syst. Zool. 34, 21–24.
Farris, J.S., 1989. The retention index and the rescaled consistency index. Cladistics 5, 417–419.
Felsenstein, J., 2004. Inferring Phylogenies. Sinauer Associates, Sunderland, Sunderland.
Foulds, L.R., Graham, R.L., 1982. The Steiner problem in phylogeny is NP-complete. Adv. Appl. Math. 3, 43–49.
Goloboff, P.A., 1993. Estimating character weights during tree search. Cladistics 9, 83–91.
Goloboff, P.A., 1999. Analyzing large data sets in reasonable times. Cladistics 15, 415–428.
Goloboff, P.A., 2013. Extended implied weighting. Cladistics. http://dx.doi.org/10.1111/cla.12047.
Goloboff, P.A., Simmons, M.P., 2014. Bias in Tree Searches and its Consequences for Measuring Group Supports. Syst. Biol. (accepted for publication).
Goloboff, P., Farris, J.S., 2001. Methods for quick consensus estimation. Cladistics 17, S26–S34.
Goloboff, P.A., Pol, D., 2002. Semi-strict supertrees. Cladistics 18, 514–525.
Goloboff, P.A., Carpenter, J.M., Arias, J.S., Miranda Esquivel, D.R., 2008a. Weighting against homoplasy improves phylogenetic analysis of morphological data sets. Cladistics 24, 758–773.
Goloboff, P.A., Farris, J.S., Nixon, K.C., 2008b. TNT, a free program for phylogenetic analysis. Cladistics 24, 774–786.
Kluge, A.G., Farris, J.S., 1969. Quantitative phyletics and the evolution of anurans. Syst. Zool. 18, 1–32.
Maddison, D., 1991. The discovery and importance of multiple islands of most parsimonious trees. Syst. Zool. 40, 315–328.
McMahon, M.M., Sanderson, M., 2006. Phylogenetic supermatrix analysis of GenBank sequences from 2228 Papilionoid legumes. Syst. Biol. 55, 818–836.
Nixon, K.C., 1999. The parsimony ratchet a new method for rapid parsimony analysis. Cladistics 15, 407–414.
Radel, D., Sand, A., Steel, M., 2013. Hide and seek: placing and finding an optimal tree for thousands of homoplasy-rich sequences. Mol. Phylogenet. Evol. 69, 1186–1189.
Ragan, M., 1992. Phylogenetic inference based on matrix representation of trees. Mol. Phylogenet. Evol. 1, 51–58.
Semple, C., Steel, M., 2003. Phylogenetics. Oxford Lecture Series in Mathematics and Its Applications, vol. 24. Oxford University Press.
Steel, M., 1992. The complexity of reconstructing trees from qualitative characters and subtrees. J. Classification 9, 91–116.
Steel, M., 2014. Tracing evolutionary links between species using mathematics. Am. Math. Monthly (submitted for publication).
Swofford, D.L., Olsen, G., 1990. Phylogeny reconstruction. In: Hillis, D., Moritz, C. (Eds.), Molecular Systematics, pp. 411–501.
Swofford, D.L., 2001. PAUP∗: Phylogenetic Analysis using Parsimony (∗and other methods). Sinauer Associates, Sunderland.