Calculating SPR distances between trees

Pablo A. Goloboff*

CONICET, Instituto Superior de Entomología, Instituto Miguel Lillo, Miguel Lillo 205, 4000 SM. de Tucumán, Argentina

Accepted 12 July 2007

Abstract

The SPR distance between two trees is the minimum number of SPR moves required to convert one tree into the other. It has been proven as an NP-complete problem. A heuristic to calculate SPR distances between trees is described. It performs favorably when compared with other existing heuristics, RIATA-HGT and EEEP. Compared with RIATA-HGT, the new method tends to produce better estimations when the trees are relatively similar, and worse estimations when the trees are very different (e.g., random trees); it produces results rather similar to those of EEEP, but orders of magnitude faster. A measure of tree-similarity based on SPR distances is proposed, obtained by calculating the minimum number of weighted SPR moves (with moves to closer nodes being less costly). The resulting measure of similarity is symmetric (i.e., \( D_{ij} = D_{ji} \), for any two trees \( i,j \)).

In taxonomic studies, the number of branch moves necessary to convert one tree into another may often be useful to express the degree of difference between them; this is known as the SPR distance between the trees. An example of the use of SPR distances in a taxonomic context is Goloboff et al.’s (2006) comparison between trees derived from separate analyses of discrete and continuous characters. The calculation of SPR distances has been proven as an NP-complete problem (Bordewich and Semple, 2005), and it has attracted attention in recent years, especially in relation to problems of horizontal gene transfer and hybridization (Maddison, 1997; Nakhleh et al., 2005; Baroni et al., 2006; Beiko and Hamilton, 2006).

None of the major programs used in systematics has so far included options to calculate SPR distances (only Component, Page, 1993; included a function to calculate Nearest Neighbor Interchange distances, using the algorithms of Brown and Day, 1984). An approximation algorithm for SPR distances was given by Bonet et al. (2006), and two heuristic methods have been published by Nakhleh et al. (2005; implemented in the program RIATA-HGT, at http://bioinfo.cs.rice.edu/phylonet) and by Beiko and Hamilton (2006; implemented in the program EEEP, at http://bioinformatics.org.au/eeep). The purpose of this paper is to describe an alternative heuristic method, implemented in the computer program TNT (Goloboff et al., 2003) since May 2005. Compared with the other heuristics, the new method often produces better estimations of the number of SPR moves between two trees.

The method

The method takes advantage of the fact that trees can be represented with a matrix \( M \) of group membership variables (Farris, 1973, 1991; Brooks, 1981; also called MRP matrices, Baum, 1992; Ragan, 1992), and that parsimony calculations under SPR can be done quickly (using techniques described in Goloboff, 1994, 1996). Then, a crude method can be obtained by creating a matrix \( M \) that represents the target tree \( T \) (with every group in \( T \) represented by a binary character with state 1 for the members of the group, and state 0 for non-members), and then using the source tree \( S \) as a starting point for SPR branch swapping. Every time a move improves the fit of \( M \) (which is to say, every time a move
makes the tree more similar to $T$), the move is recorded and a counter increased. An advantage of this is that the highly efficient algorithms for SPR branch swapping already existing in TNT can be coopted for this purpose. A similar method had been implemented, as early as 1982, in the mainframe program Physis (Farris and Mickevich, 1982), to calculate “TBR distances” (i.e., using the TBR instead of the SPR swapper; J. S. Farris, pers. comm.).

$M$ will have only perfectly congruent characters, which has three advantages: (1) SPR will easily find an optimal tree; (2) the incremental-optimization algorithms of Goloboff (1996) will require reoptimization of low numbers of nodes; and (3) the SPR process can be stopped as soon as tree length equals the number of characters in $M$ (i.e., the length of $T$). To further increase speed, a constraint for monophyly can be created using the strict consensus of $S$ and $T, C$. Because of the way in which constraints are implemented in TNT (see documentation), constraining nodes for monophyly makes it possible to complete swapping faster (e.g., when the nodes around the node to be pruned are required to be monophyletic, the pruning does not even need to be effected). This method provides a coarse estimate, but it can be done quickly. Using the SPR branch-swapper of TNT (for comparisons between trees with 84 taxa and an average of 30 moves required for the conversion, as reported on Fig. 2A) completing SPR to convert $S$ into $T$ takes on average about 5 ms. Different estimations can be obtained by repeatedly using $S$ as starting point for SPR swapping, but randomizing the sequence with which nodes are clipped and reinserted (that is, “pruned-and-regrafted”); the larger the number of SPR sequences that are tried, the greater the likelihood of finding the optimal $S \rightarrow T$ path. The number of possible paths $S \rightarrow T$, however, can be quite large, and most of the clip sequences will fail to find the optimum path.

The method can be made more precise, without sacrificing speed, by taking into account that no move will violate the groups of $C$, and that the moves needed to resolve each of the polytomies created by strict-consensizing the two input trees can be counted independently and subsequently combined. The latter property was proven by Baroni et al. (2006), and also used in Beiko and Hamilton’s (2006) method. Although the moves needed to resolve each of the polytomes nodes in $C$ can be counted independently, there is no need to repeat the SPR procedure for each polytome node—a continuous SPR sequence can be used anyway. Using a continuous SPR sequence may find an optimal path to resolve some polytomy in $C$, and a suboptimal path to resolve another. However, by referring to the consensus $C$, it is possible to identify the group that a given move is solving, so that the optimal paths needed to resolve each polytomy of $C$ can be identified, subsequently combining them in one globally optimal path.

To illustrate this (and the associated bookkeeping), consider the case of the two trees in Fig. 1. The two trees differ in just two rearrangements; $S$ was obtained by modifying $T$, moving $B$ to be sister of $F$, and moving $G$ to be sister of $J$. Tree $C$ has two nodes, $C_1$ and $C_2$. The two paths shown in Fig. 1 have three and four moves. Neither path is optimal, but considering the two paths together, it is possible to deduce that paths with only two moves must exist. For this, every node in the current tree (source tree, initially, but subsequently modified) that corresponds to a node in $C$ must be marked (see Fig. 1). When a move improves the fit to $M$, the most recent marked common ancestor (MCA) of the pruned node and the node where it is to be regrafted must be identified (e.g., in the first, leftmost move of path $A$, the most recent MCA of $E$ and $F$ in $S$ corresponds to node $C_1$). In that way, separate counters (instead of a single counter for the overall number of moves) are used for counting the moves resolving each of the polytomies in the consensus $C$. Node $C_1$ corresponds to the MCA of just one move in path $A$, which implies that it is possible to resolve the polytomy of node $C_1$ with a single move. Path $A$ guarantees that node $C_2$ can be resolved with two moves or less. Path $B$ is a poorer one for node $C_1$, but better for $C_2$: $C_2$ corresponds to the MCA of a single move in path $B$. Thus, although paths $A$ and $B$ have three and four moves each, they together determine a two-move path $S \rightarrow T$. By calculating the consensus of $S$ and $T$, the problem of identifying the shortest path $S \rightarrow T$ can be greatly simplified; as long as each of the polytomies in $C$ is optimally solved in one of the paths (i.e., in one of the randomized sequences of clipping), then the optimal solution will be identified. This will work better to the extent that $C$ is better resolved; polytomies of many nodes will be harder to solve, especially when many moves are required to solve them.

A closer examination of Fig. 1 reveals the source of (some of) the failures to find optimal paths. As $E$ and $F$ form a monophyletic group in target tree, if $E$ is moved first, then placing $E$ as sister of $F$ will improve the fit to $M$, saving one step in the variable that represents group (EF). The fit to that same character (and others as well) would have also been improved if taxon $B$ had been moved first, to the base of the tree—this would have saved three steps, making three of the groups in $C$ better resolved; polytomies of many nodes will be harder to solve, especially when many moves are required to solve them.

A similar method had been implemented, as early as 1982, in the mainframe program Physis (Farris and Mickevich, 1982), to calculate “TBR distances” (i.e., using the TBR instead of the SPR swapper; J. S. Farris, pers. comm.).
Therefore, if \( a_0 \) is initially specified by the user, only prunings for which \( \Delta L_p = a_0 \) can save \( a_0 \) steps (thus making evaluation of alternative locations of the pruned node unnecessary). All the clippings will be evaluated in the first cycle, and for the next one, \( a_1 \) will be used as threshold (setting \( a_1 \) to the maximum value of \( \Delta L_p \) in the previous cycle that was below \( a_0 \), so that setting \( a_0 \) to a large value does not slow down calculations beyond necessity). This will continue, with successively smaller values of \( a \), until either \( a_n = 0 \) or tree length equals the number of characters in \( M \).

When the paths \( S \rightarrow T \) are evaluated using stratification, the results are usually improved significantly. For the trees in Fig. 1, different unstratified SPR sequences produce paths with two, three and four moves (with approximate frequencies of 0.25, 0.50 and 0.25, respectively), while stratified SPR (with \( a_0 = 3 \)) always produces paths with two moves. In some rare cases, however, stratification may worsen results; these tend to be cases where \( C \) is very poorly resolved and many moves are required to complete \( S \rightarrow T \); for these cases, the simple (unstratified) procedure sometimes finds shorter paths. In the implementation in TNT, it is also possible to optionally use combined evaluations, alternating both methods. Although larger values of \( a_0 \) tend to produce better estimations, sometimes a lower value may (as with any heuristic method!) find a shorter \( S \rightarrow T \) path.

If the paths are to be interpreted as indicating events of horizontal gene transfers, then it may be desirable to avoid moves that imply a time violation, i.e., moves where a node is moved to be sister group of one of its own ancestors, which—unless additional nodes in the tree are postulated, see discussion in Baroni et al. (2006)—would imply that the ancestor receives genes from its own descendant. For individual moves, this can be easily done in the context of the present method, because the moves are being effected on actual trees.

Fig. 1. Two trees (source and target), and two possible paths between them. See text for explanation.
Several moves, however, may imply a time violation when combined (see Baroni et al., 2006), making the postulation of additional nodes unavoidable.

**Comparison with RIATA-HGT**

As a first test, the method was compared with RIATA-HGT, creating pairs of trees by means of different random addition sequence Wagner trees (Farris, 1970), for the data set of Goloboff (1995). This gave sets of trees with considerable differences, but not as different as random trees. The timings reported in this section correspond to a machine with a 2.4 GHz Celeron processor, running under Windows XP. The numbers of moves found by TNT (the best value of 150 stratified replications with \( a_0 = 100 \), 150 stratified replications with \( a_0 = 5 \), and 150 unstratified replications) and the number of moves found by RIATA-HGT for these pairs of Wagner trees are plotted in Fig. 2(A). Points on the diagonal indicate that TNT and RIATA-HGT found the same value for the corresponding pair of trees. On average, TNT found paths with one less move than RIATA-HGT (29.00 versus 30.03 moves, respectively). The time used by both programs in most of the runs was comparable (taking into account that RIATA-HGT was running under Java), but some of the RIATA-HGT runs took much longer than others (10% of the RIATA-HGT runs took from 60 to 146 s, as opposed to a uniform 2–3 s in TNT).

As an additional comparison between TNT and RIATA-HGT on binary trees, pairs of trees (with 60–100 taxa) were created by generating two copies of a random tree and then rearranging (at random) some branches of one of them (with three to 20 SPR moves, to locations up to 30 nodes away). This produced the

![Fig. 2. (A) Values of RIATA-HGT as a function of the TNT values, on pairs of Wagner trees with a different random addition sequences (values above the diagonal indicate a poorer performance of RIATA-HGT). (B) Values of RIATA-HGT (white triangles) and TNT (black circles) as a function of the number of moves used to produce the modified tree ("true" value), on simulated trees (values above the diagonal indicate cases in which the minimum was not found). (C) Values of RIATA-HGT as a function of the TNT values, on pairs of random trees (values below the diagonal indicate a poorer performance of TNT). (D) Values of EEEP as a function of the TNT values, on pairs of Wagner trees with different random addition sequences (values above the diagonal indicate a poorer performance of EEEP).](image)
results shown in Fig. 2(B). The y-axis shows the number of moves found by TNT (black circles) and RIATA-HGT (white circles), with the numbers of moves used to produce the modified tree on the x-axis. The shortest path between the two trees, in this case, cannot be longer than the path used to generate the rearranged tree, so that the many points above the diagonal show that RIATA-HGT often produces serious overestimations of SPR distances. TNT, again, outperforms RIATA-HGT, with fewer points above the diagonal, and finding (on average) paths with 1.4 fewer moves than RIATA-HGT.

One of the reasons for the somewhat lower performance of RIATA-HGT may be that the method is based on calculating first an agreement subtree (subsequently adding terminal taxa to it), but there are cases in which the agreement subtree has few taxa even when the input trees are rather similar and require few SPR moves for interconversion (see Goloboff et al., 2006 for an example). On the other side, as implied in the description of the algorithms of the preceding section, when source and target trees share very few groups and their consensus has larger polytomies, the method described here is expected to perform worse; for pairs of random trees, TNT produces worse estimations than RIATA-HGT (Fig. 2C).

It is reassuring that both RIATA-HGT and the current method provide accurate estimations for low numbers of moves. For the problem of gene transfer, this may be enough to make the algorithms entirely reliable: gene transfer is probably not a very common event. On the other hand, if the purpose is establishing taxonomic stratification in more levels) should be used when precision is desired under those circumstances.

Comparison with EEEP

As a second test of the method implemented in TNT, it was compared with Beiko and Hamilton’s (2006) EEEP program. The recommended option in that program1 works by attempting all the SPR rearrangements starting from source tree $S$, and keeping all the resulting trees (one SPR move apart from $S$) that are more similar to $T$ than the original (with tree distance calculated as number of groups of $T$ absent from current tree). Each of these trees is then subjected to SPR, and every one of the trees (now two SPR moves apart from $S$) more similar to $T$ is retained. Eventually, some of the trees produce tree $T$. Like TNT, EEEP also calculates the strict consensus of $S$ and $T$, and avoids moves violating the groups of $C$. Note that if, instead of saving only those trees that are more similar to $T$, all the trees resulting from SPR moves are saved at each stage (which is optional in EEEP), then an incredibly time-consuming but necessarily correct solution is obtained.

Differences in how the distance to $T$ is measured aside, EEEP and the present method are rather similar. The main differences are in:

1 **Strategy.** The strategy used by TNT might be compared with doing multiple random addition sequences for parsimony searches (without saving equally parsimonious trees), and the strategy used by EEEP might be compared with doing a single search saving as many equally parsimonious trees as RAM will allow. The latter is known to be a much more time-consuming strategy (e.g., Goloboff, 1999; Davis et al., 2004).

2 **Implementation.** TNT makes use of parsimony techniques, so that every rearrangement can be evaluated very quickly, in constant time (using algorithms of Goloboff, 1994, 1996). EEEP instead compares every group in the current tree with every group in the target $T$. The work needed for evaluating every tree thus increases rapidly with the numbers of taxa.

3 **Treatment of root.** EEEP considers the source tree $S$ as unrooted, thus sometimes finding paths shorter than can be realized without allowing for rerooting. Depending on what the two trees being compared represent, considering alternative rootings may or may not be desirable.

Given the similarities and differences between the two methods, it is not surprising that EEEP produces results very similar to those found by TNT, but using about 2000 times longer (the timings reported below correspond to a 3.0 GHz Pentium IV machine, running under Suse Linux 9.1, for both TNT and EEEP). For the present test, 20 pairs of trees with 30, 40, 50, 60 and 70 taxa were used (for a total of 100 pairs of trees); the results are plotted in Fig. 2D. The trees were Wagner trees for random subsets of the corresponding number of taxa, out of the 84-taxon matrix of Goloboff (1995). The average time used by TNT for calculating distances between these trees (using 200 rounds with $a_0 = 5$) was 0.31 s. Many of the EEEP runs (36%, overall) had not finished after 15 min or more2, especially for larger

---

1 Beiko and Hamilton (2006) call this option (as opposed to the option where all the trees resulting from SPR rearrangements are retained) “ratchet”. I do not know whether they call it “ratchet” by analogy with Nixon’s (1999) algorithm, because they neither cite nor mention Nixon (1999).

2 The code of EEEP was modified so that the time reported was for the calculation of the minimum distance only. The original version of the program, subsequent to finding moves in shortest paths, tries alternative sequencings of those moves, in search of paths that do not violate time constraints; this is a very time consuming part of the process, and was not counted here.
numbers of taxa (for 70 taxa, only 25% of the cases finished; for 30 taxa, every single one did). Even if these runs had finished and produced some result, the difference of speed in favor of TNT would have been about 2100 times. For the 64 runs in which EEEP finished, TNT found shorter paths in 28% of the cases, while EEEP found shorter paths in 8% of the cases. Despite some differences, these were much less marked than in the case of RIATA-HGT.

In the case of random trees, only 20 taxa could be tested, due to time constraints. EEEP was also frequently unable (20% of the cases) to finish calculations within 15 min; it found, a third of the times, shorter paths than TNT, but checking some of the paths showed that they could not be realized without rerooting—so that the difference may have to do with differences in how EEEP considers rerooting. In the cases checked, trying alternative rootings of the source tree S (which in some cases may not be desirable) quickly produced paths as short as those of EEEP. This shows that—differences in the treatment of rooting aside—EEEP and TNT produce essentially the same results, for both relatively similar trees and random trees.

**Polytomous trees**

SPR moves are defined only for dichotomous trees. In the case of polytomous trees, the implementation in TNT considers, by default, the SPR distance between the dichotomous resolutions of S and T that are closest to each other. Polytomies in T simply mean that M will have fewer characters (and that several alternative resolutions will match the polytomies in T). Polytomies in S have to be resolved, but this is done easily, by creating a constraint with the polytomous tree S, and then submitting it to SPR to maximize the fit to M under those constraints (and without counting the number of moves required for this). The procedure described above can subsequently be applied, just like before, to the tree produced by SPR constrained to have all the groups in S. When polytomies are treated in this way, the distance between a completely unresolved bush and a dichotomous tree is always zero. In other words, more than general topological similarity, the SPR distance for polytomous trees measures the number of incompatible differences between both trees (i.e., none, in the case of any tree versus a complete bush). This may be useful under many circumstances (e.g., in studies of congruence between data sets, as in Goloboff et al., 2006); it is the default method implemented in RIATA-HGT and TNT.

Alternatively, if a measure of topological similarity is desired, it would perhaps be possible to consider the number of branch moves required to convert one tree into the other, expanding the repertoire of possible moves to include collapsing and resolving of branches. TNT optionally implements an approximation to this, which usually produces the correct value using only a little extra time. The procedure is as follows (note that steps 1 and 2 are part of the procedure described before, for comparing the closest resolutions of each polytomy):

1. Search with constraints from source tree S to produce a binary tree B with a fit as good as possible within those constraints.
2. Submit B to SPR, and record the number of moves needed to produce tree P (possibly identical to T), with perfect fit to M, as described before.
3. Count the number of groups in B that are present in P and fulfill one of the conditions (a) or (b) as follows:
   a) have “synapomorphies” provided by M (i.e., changes from 0 to 1 in the group membership variable) and are absent from S (these are groups that had to be resolved from polytomies in S),
   b) have no “synapomorphies” provided by M and are present in S (these are groups that have to be collapsed in order to match T).
4. Add the number obtained in (3), which indicates moves required to collapse/resolve nodes, to the number of moves obtained in (2).

The RIATA-HGT program has the option (with the –u flag) to use the trees without “contracting and refining” (i.e., collapsing and resolving), and this sometimes produces the same value as the approximation just described. However, the RIATA-HGT algorithm has asymmetries when one tree is a resolution of the other. From (A(B(BCD))) to (A(BCD)), a single move is (correctly) counted with the –u option, but from (A(BCD)) to (A(B(BCD))), the program reports zero moves. This implies that trees with different unresolved groups cannot be analyzed with RIATA-HGT: (A((BCD)(E(FG)))) and (A(B(CD))(EF)) are a symmetric pair of trees, each with a different resolution of the trichotomy of the other tree; one move is needed to resolve/collapse each of the trichotomies. Therefore, the minimum number of moves (when moves to collapse/resolve branches are counted) is two, which is the value calculated with the approximation described above. In contrast, RIATA-HGT (with the –u option) reports one move (an impossibly low value) to convert any of the trees into the other; the program seems to be using the criterion of comparing the closest resolutions for one of the three taxon groups, and the criterion of counting moves needed to resolve/collapse for the other.

**Weighted moves**

If the number of SPR moves is to be considered as indicating tree similarity (or the degree of conflicting differences in groupings, in the case of polytomous trees), then not all moves can be considered as equiv-
alent. Clearly, a move to a far distant alternative location implies more “difference” than a move to a closer position. The moves then can be weighted, according to their distance\(^1\). The distance of a move can be calculated as the degree of the polytomy created by consensing the trees before and after the move, minus 2. In TNT, a move to \(n\) nodes away, can be given a weight \(n/n + k\) (where \(k\) is a constant that determines the difference in cost for different distances; when \(k = 0\), all moves cost the same; as \(k\) increases, there is a greater difference in cost for moves to different distances). To make the values more easily interpretable, they can be rescaled dividing by a constant factor \(f\), such that the longest possible move (for \(t\) taxa, \(t - 3\) nodes away) has a weight of 1. The weight for a move \(n\) nodes away then becomes \(w = n/f (n + k)\) (where \(f = t - 3/k + t - 3\)). This provides a measure of tree similarity more meaningful than just counting the number of moves regardless of their distance.

Finding the shortest weighted path is harder than finding the shortest unweighted path (already an NP-complete problem). In the implementation of TNT, when alternative positions of the pruned subtree all imply the same improvement in the fit to \(M\), the position that minimizes the weighted cost is chosen. Given the choice between improving fit to \(M\) or minimizing move cost, improving fit to \(M\) is always preferred. Thus, the method will effectively find shortest weighted paths for low values of \(k\) (when there is not a very big difference between moves), but may more easily fail to do so for high values of \(k\). Perhaps other modifications of the algorithm for unweighted moves will produce more effective means for finding shortest weighted \(S \rightarrow T\) paths.

**Acknowledgments**

I thank M. Mirande, L. Nakhleh, D. Pol, C. Szumik, L. Táher, two anonymous reviewers, and Associate Editor Kurt Pickett for discussion, comments on the manuscript, and/or directing me to additional literature on SPR distances. This work was supported by the ANPCyT (PICT 12605 FONCYT-BID), CONICET (PIP 02567), and Consejo de Investigaciones de la Universidad Nacional de Tucumán, Argentina.

\(^{1}\)A scheme for weighting moves has been proposed also, in the context of Nearest Neighbor Interchange (NNI) distances, by DasGupta et al. (1997), but it concerns the “weights” (e.g., group supports) of the individual branches to be eliminated during NNI. In the case of NNI, the weighting by distance employed here is neither applicable (all NNI moves are just local rearrangements) nor necessary (if two trees differ in having a taxon alternatively placed at two far distant locations, then more NNI moves will be required to connect the two trees).

**References**


