WILEY-BLACKWELL

Cladistics (2016) 1–18

Cladistics

10.1111/cla.12160

TNT version 1.5, including a full implementation of phylogenetic morphometrics

Pablo A. Goloboff[a,*] and Santiago A. Catalano[a,b]

[a]Unidad Ejecutora Lillo, Consejo Nacional de Investigaciones Cientfficas y Técnicas, Miguel Lillo 251, 4000 S.M. de Tucumán, Argentina;
[b]Facultad de Ciencias Naturales e Instituto Miguel Lillo, Universidad Nacional de Tucumán, Miguel Lillo 205, 4000 S.M. de Tucumán, Argentina

Accepted 29 February 2016

Abstract

Version 1.5 of the computer program TNT completely integrates landmark data into phylogenetic analysis. Landmark data consist of coordinates (in two or three dimensions) for the terminal taxa; TNT reconstructs shapes for the internal nodes such that the difference between ancestor and descendant shapes for all tree branches sums up to a minimum; this sum is used as tree score. Landmark data can be analysed alone or in combination with standard characters; all the applicable commands and options in TNT can be used transparently after reading a landmark data set. The program continues implementing all the types of analyses in former versions, including discrete and continuous characters (which can now be read at any scale, and automatically rescaled by TNT). Using algorithms described in this paper, searches for landmark data can be made tens to hundreds of times faster than it was possible before (from T to $3T$ times faster, where T is the number of taxa), thus making phylogenetic analysis of landmarks feasible even on standard personal computers.

Since the first non-beta release of TNT in 2003 (see Goloboff et al., 2004; Giribet, 2005; Hovenkamp, 2004; Meier and Ali, 2005), the program continued being improved on several fronts. Most of the general options for a full phylogenetic analysis were already present by 2008, when the program was made freely available, subsidized by the Willi Hennig Society (Goloboff et al., 2008). Many options and facilities for specific types of analyses or calculations have continued being added to the program after 2008, such as the option to incorporate and automatically test taxonomies (Goloboff and Catalano, 2012), the extension of implied weighting to take into account missing entries and average amounts of homoplasy in sets of characters (Goloboff, 2013), new search routines for difficult data sets (Goloboff, 2014, 2015), or the implementation of algorithms to identify wildcard taxa (Goloboff and Szumik, 2015).

*Corresponding author.
E-mail address: pablogolo@yahoo.com.ar

Other programs compete with TNT for the analysis of sequence data (especially POY: Wheeler et al., 2015; RaxML: Stamatakis, 2014; MrBayes: Ronquist et al., 2012) and are widely used. However, no other program includes TNT's array of options for morphological data sets, in terms of optimality criteria, analytical options, diagnostic facilities and data types. In this last sense, one of the most important recent additions to the program has been the incorporation of a new type of character: landmark configurations in the original form of two- or three-dimensional (2D, 3D) coordinates. The approach was first described by Catalano et al. (2009, 2010), who discussed the rationale for the method under the parsimony criterion. Although there had been attempts to use data from geometric morphometrics in a phylogenetic context (Rohlf, 2002; Lockwood et al., 2004; González-José et al., 2008; Klingenberg and Gidaszewski, 2010), the method of Catalano et al. (2010) is the only one that establishes a clear, well-justified bridge between geometric morphometrics and conventional parsimony analysis. In other

types of characters, evaluating a tree requires identifying the optimal ancestral conditions, those conditions that minimize the amount of difference across all branches of the tree (e.g. Farris, 1983, 2008). In exactly the same way, the ancestral conditions for a single landmark will be the positions (coordinates) such that the amount of difference across all branches of the tree is minimal; these will be found independently for each of the points forming a configuration, thus indirectly determining an ancestral shape. The measure of difference to minimize is, as in the optimization of additive characters (Farris, 1970), the lineal, physical distance between points (i.e. the square root of the sum of the squared differences in the two or three axes; note that this distance is immune to rotations or translations of the system of coordinates). Finding the optimal ancestral coordinates for a given tree is a difficult problem, akin to the problem of Steiner trees (Brazil et al., 2014), a well-known computationally difficult problem. Evaluating tree-scores or finding optimal ancestral conditions is also a difficult problem in other criteria for phylogenetic evaluation, such as maximum likelihood or unaligned sequences (as discussed by, for example, Swofford et al., 1996; Wheeler, 1996), for which only heuristic algorithms exist. Goloboff and Catalano (2011) proposed algorithms to estimate ancestral landmark coordinates, which were implemented and tested in TNT. These algorithms are heuristic, but in practice produce good approximations.

The implementation of landmark characters in TNT, however, was still rather incomplete. It was possible to evaluate any given tree using the algorithms of Goloboff and Catalano (2011), and in combination with the scripting language of TNT (as described in, for example, Goloboff et al., 2008: 783–785) it was possible to do tree searches (e.g. Catalano et al., 2015; Perrard et al., 2015). However, these searches were very limited, for two reasons. First, many of the rearrangement and search methods included in TNT for other types of characters are too involved to be easily emulated with scripts; scripts only allow basic search algorithms such as random addition sequence Wagner trees (RAS) and branch-swapping. Second, and even worse, using scripts required that every tree be evaluated anew, which is equivalent to doing a tree search using a full down-pass optimization in a standard character to evaluate each tree rearrangement attempted. Successful tree-search algorithms require that the scores of the rearrangements to a tree are calculated without doing a full optimization (an issue discussed by Goloboff, 2015: 211–213; and papers cited therein, for standard characters). Goloboff and Catalano (2011: 50) had already noted that in the case of landmarks even simple data sets could require days of CPU time when using scripts, and stressed the need for faster tree-search algorithms implemented natively.

These problems have now been overcome, with much faster methods (described in this paper) for approximating tree scores during searches. These methods are between T to $3T$ times faster (where T is the number of taxa) than the methods previously used. Thus, essentially all the search options previously available in TNT (see Goloboff et al., 2008; Goloboff, 2014) for other types of characters can now be applied to landmark characters as well. These additions, together with recent improvements and additions in recent years, are deemed important enough to warrant switching from version 1.1 (since 2006) to version 1.5. Analysis of both continuous (which no longer require to be input in a scale of 0–65, and can be automatically rescaled by TNT) and discrete characters continues to be possible, and landmark data are easily incorporated into matrices of standard characters. For the basic commands and options available in the program, the reader is referred to Goloboff et al. (2008). This paper focuses on the methods used to quickly estimate tree scores for landmarks during searches, and on the general implementation of these methods in TNT, to facilitate using these new options in the best possible way.

Importing/exporting landmarks

TNT can now import files in the TPS format (http://life.bio.sunysb.edu/morph/soft-tps.html), one of the most widely used formats in geometric morphometrics. Typical TPS files contain only landmark configurations from a single structure, because geometric morphometric studies usually deal with the shape of a single structure at a time. For inferring phylogenies, however, it is necessary to consider as much evidence as possible, and TNT easily allows combining several TPS files into one TNT data file. With the *dmerge* command (or with the menu option **File/MergeImport/ImportTPS Files**), each configuration is placed in a block of its own, and the corresponding character is automatically named as the source file containing the configuration.

TPS files will most commonly contain configurations aligned by the least-squares Procrustes method (Rohlf, 1990; Rohlf and Slice, 1990). As discussed by Catalano et al. (2010), other types of alignments may be more appropriate for phylogenetic analysis, especially those minimizing linear distances (see Larsen, 2008). TNT therefore allows the pairwise realigning of configurations against a specified reference taxon, using either RFTRA (Siegel and Benson, 1982; Rohlf and Slice, 1990) or an algorithm that minimizes linear distances heuristically (by translating and rotating shapes); these options are available with the command *lmrealign*, or the menu choice **Data/EditData/LandmarkAlignment**.

The method for aligning landmark configurations taking into account the structure of a tree (Catalano and Goloboff, 2012) can be used to further increase the fit of configurations to the final tree produced by a search based on a static alignment (tree searches in TNT can only be done for the static, fixed alignments). These tree-based alignments were previously available only for 2D data, but version 1.5 incorporates the same approach also for landmarks in 3D. Note that for data sets with multiple configurations, scores before and after the tree alignment are guaranteed to improve only when the factors (to make different configurations comparable) are fixed (defined by the user), as the automatic factors may otherwise change after alignment (producing an apparently larger score, even if the alignment better fits the tree structure).

The new command *lmbox* allows saving simple diagrams of the ancestral configurations; Fig. 1 is an example, for the pelvis in the common ancestor of hyaenids in a tree for Martín-Serra et al.'s (2014) data set. The *export* command also allows saving the coordinates in TPS format, including the shapes reconstructed for the internal nodes of trees, for better graphics or for more detailed analyses with specific programs.

Tree searches: general approach

The core of most search algorithms is in branch-swapping. The rearrangements during branch-swapping are generated by clipping a clade off the main tree, and reinserting it back (possibly rerooted) at the available locations. For standard characters (categorical and one-dimensional: continuous, additive, non-additive or step-matrix characters), both the lengths for each character in the subtrees and the length increment produced by reinserting the clipped clade can be calculated exactly with very little work. In the case of landmark characters, as discussed above, the tree scores cannot be calculated exactly, even for a single tree.

The new version of TNT uses, in the case of landmarks, a first estimation of the length increment produced by reinserting the clipped clade, approximate and based on a looser (less precise) estimation than the one used for calculating the score of an entire tree. For increasing numbers of taxa, this estimation uses (for each landmark) an approximately constant time, as it is based on considering only the positions of the points delimiting the branches to be reconnected.

These quick initial calculations may overestimate tree length, thus leading to rejection of rearrangements that should in fact be accepted. This is prevented by leaving some margin for errors—trees are rejected only when their estimated score exceeds that of the best trees by a certain factor. To improve precision and provide coherence, when this quick estimation indicates that a rearrangement may be preferable to the best tree found so far, the tree length is recalculated by means of a full optimization with the more intensive algorithms described by Goloboff and Catalano (2011). In this way, the scores reported during the search are always the same scores obtained with a recalculation (anew) after the search finishes (e.g. if the trees resulting from the search are saved to a file and subsequently read into TNT in a new session). The implementation in TNT takes special precautions to ensure that the scores calculated during and after the search are (for the same parameters) identical; the details are explained in Appendix 1.

To the extent that the quick initial calculation allows rejecting many rearrangements without the need to use the more intensive algorithms, the search will be speeded up.

For all the descriptions that follow, keep in mind that tree-length calculations during searches only consider binary trees.

Quick estimation of tree-scores during searches with four-point Steiner trees (FPSTs)

The algorithms described by Goloboff and Catalano (2011) to approximate optimal positions for the ancestral landmarks on a given tree are based on two

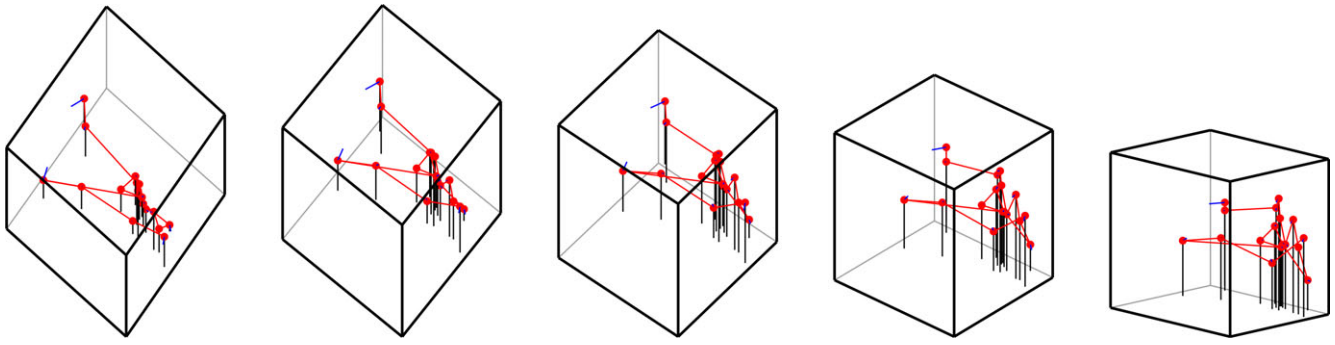

Fig. 1. Example of image saved as SVG file, for an ancestral node, reconstructed for a 3D landmark configuration, shown with different rotations and tilts.

stages: first, using a grid to calculate the costs at a series of possible fixed positions at the ancestral nodes (as in a Sankoff optimization; for details see Goloboff and Catalano, 2011), and second, iteratively calculating Fermat points for each internal node. Although the second stage is iterative and must possibly cycle through all or most of the internal nodes several times, it takes a much shorter time than the first. The initial calculations are akin to optimizing a Sankoff character with possibly hundreds of states; the times required for Sankoff optimization increase with S^2 or more, where S is the number of states (Clemente et al., 2009 proposed algorithms for Sankoff optimization that use less time than S^2, but these algorithms have not been implemented in the present version of TNT). The initial, grid-based calculations can be improved by using *nested* grids, in which a smaller grid is created around the region where the point is located by the first initial grid (see Goloboff and Catalano, 2011).

For a faster approximation to the differences in score that will result from joining the two subtrees during branch-swapping, TNT uses the distances between the optimal positions of the points delimiting the branches to be connected, as illustrated in Fig. 2, with a method based on FPSTs. Consider the case where two branches are to be connected; these branches are delimited by nodes *A–D*. Let d_{ij} be the distance between the point position (for the landmark in question) at node i and the point at node j [i.e. $((i_x–j_x)^2 + (i_y–j_y)^2)^{1/2}$ in the case of 2D, and $((i_x–j_x)^2 + (i_y–j_y)^2 + (i_z–j_z)^2)^{1/2}$ in the case of 3D]. The points for the landmark must be positioned, in both subtrees, such that the sum $\sum d_{n,ancestor}$ for all the tree nodes n that have an ancestor (i.e. excluding the nodes marked as r in Fig. 2 from the calculation) is minimum. The connection of the two branches will require creating two new nodes, x and y, and the length increment resulting from joining the two branches can be approximated as the length of the Steiner tree given the topology $((AB)(CD))$, where coordinates for points *A–D* are fixed, minus the length of the originally existing branches joining A with B, and C with D. More precisely, the resulting length increment can be estimated as $\delta L = L_2–L_1$, where $L_1 = d_{AB} + d_{CD}$, and $L_2 = d_{Ax} + d_{Bx} + d_{xy} + d_{Cy} + d_{Dy}$ (after the positions for points x and y that minimize L_2 are calculated).

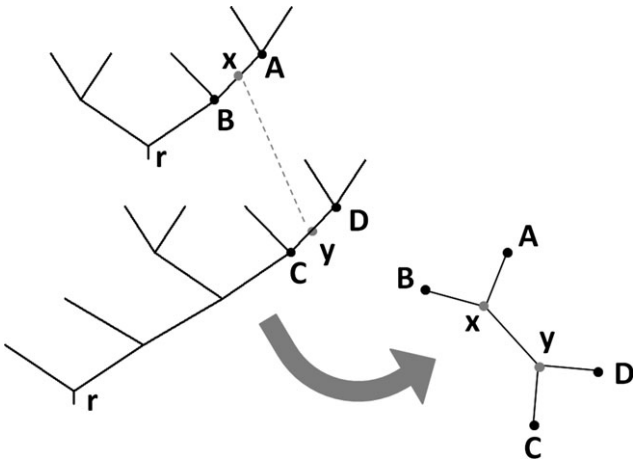

Fig. 2. Method of 'four-point Steiner trees', used to approximate the score during branch swapping. See text for explanation. A similar method is used to test taxon insertions when building Wagner trees.

The length increment thus calculated is of course not exact, because points beyond *A–D* may require repositioning once the two subtrees are connected, but it is a good approximation, and it can be calculated very quickly, with constant time for each landmark. Note that when one of the two subtrees is a single taxon, or one of the descendant nodes (A or D) is a terminal with a missing entry, the problem reduces to a three-point (instead of a four-point) Steiner tree, which is simpler (the coordinates of the middle point can be found exactly with Torricelli's geometric calculations). If both A and D have missing entries then the problem reduces to a two-point problem, and the length increase is simply $\delta L = d_{BC}$ (a similar reasoning applies when one of the two subtrees is a single terminal, and the descendant node of the other subtree has a missing entry for the landmark).

To facilitate calculations, the same FPST process is used to calculate length decrements when the tree is split in two. Thus, the rearrangements to be accepted are those for which the length increment when joining two branches is smaller than the length decrement when joining the two branches at the original positions, or when $\delta L_{\text{join}} < \delta L_{\text{split}}$.

As stated above, the points at every node n of the separate subtrees must be placed such that $\sum L_{n,ancestor}$ is minimum. To further save time, those positions are simply calculated by taking the optimal positions for the entire tree, and then iteratively recalculating the Fermat points for all the internal nodes of the two subtrees after division (recall that the iterative phase of tree-length calculations takes a much shorter time than the grid-based phase; see above). Thus, the Sankoff grid is used only for calculating point positions in the entire tree or when rechecking length of a good candidate. The optimal point positions for the entire tree are buffered separately, copied anew and iteratively modified every time the tree is split in two. When a cycle of tree-division and attempted reinsertions results in no improvement and the clipped clade is to be reinserted back at the original position (and rooting), nothing needs to be done; if the clipped clade is reinserted at a different location (or a different rooting), then the point positions for the entire tree are

calculated again first by using a grid, then iteratively recalculating Fermat points.

Error margin

To prevent the possibility that overestimation of tree scores with the shortcuts described above rejects some trees that should have been accepted, a margin of error can be introduced during the search. Recall that the trees to be rejected during branch-swapping were those for which the estimated increase of length when joining the subtrees was greater than the estimated decrease when clipping the original tree into two; that is, when $\delta L_{join} > \delta L_{split}$. To allow an error margin of E (where $0 \leq E \leq 1$), the trees can be rejected only when $(1–E) \times \delta L_{join} > \delta L_{split}$. A tree that passes this relaxed criterion is considered a good initial candidate and re-checked with a full optimization to decide whether the rearrangement is to be accepted. Note that when $E = 1$, all the trees are always accepted as good initial candidates, and (barring minor implementation details) the times needed to complete a branch-swapping search are the same as in the scripts that had to be used in earlier versions of TNT. As the error margin is decreased, more trees are rejected on the basis of the initial estimation, and the number of rearrangements that the program can examine per unit time is much larger. The possibility exists, of course, that some rearrangements that should have been accepted are rejected. The error margin thus should be selected carefully. In our experience with numerous data sets, values of $E = 0.15$ normally suffice (when used in conjunction with the incremental Fermat recalculations described below) for no good rearrangements to be missed during the search. Note that this value of error E is used to multiply the score increase estimated as δL_{join}. An alternative would be to use an absolute value of error margin (such that a constant quantity is subtracted from the estimated score), but in our experience this type of error margin seems much harder to adjust to useful values for different data sets (especially when the data sets have different numbers of landmarks/configurations).

During branch-swapping starting from RAS trees, the initial stages of the search easily find many rearrangements that represent an improvement. During this stage, therefore, the use of an error margin is counterproductive, slowing down the program and making it take longer to achieve comparable lengths. It is therefore possible to use changing error margins, such that the error margin increases as the search progresses; using the highest error margin at the end decreases the chances that some tree in the TBR (or SPR) neighbourhood of the final tree will actually be a better tree. Although branch-swapping is an open-ended algorithm and it is therefore difficult to predict how much longer it may take to complete branch-swapping, a good estimator is obtained by calculating the proportion of clippings C done to the tree without having found any better points of reinsertion, relative to the total number of branches in the tree (for T taxa, $2T–3$). Thus, a good estimator of the proportion of swapping completed is $P = C/(2T–3)$. As P reaches certain prespecified thresholds, then, the error margin can be increased. The values of error and the proportions of swapping completed can be set by the user (in up to five stages). Keep in mind, however, that (i) setting E too high during the initial stages of the search slows down branch-swapping unnecessarily (because better rearrangements are still likely to be found during these initial stages); and (ii) setting the highest (final) value of E too late in the swapping will slow down the search as it will require that a high proportion of the rearrangements are evaluated twice (first with the lower error margin, and then again with the higher). Although there is a lot of variation in the times needed to complete branch-swapping for switching to the final error margin at different proportions of the swapping completed, there are some small differences that corroborate the reasonings (i) and (ii) just given. The error margin is set with the *lmark errmarg* command; to set several stages, several values separated by a dash must be specified to indicate the errors (as percentages, the initial one is always 0 and needs no specification), followed by a slash and the percentages of swapping to switch to the next error margin. As example, in the default settings of TNT, the error margin increases in three stages, beginning at $E = 0$ until $P = 0.15$, switching to $E = 0.05$ when $P = 0.15$, and to $E = 0.15$ (its final value) when $P = 0.66$. These settings are explicitly specified with *lmark errmarg 5-15/15-66*.

When the tree score is rechecked with a full optimization, the estimated score difference δL_{join} can be compared with the actual difference, so that the error actually produced by the approximation, or observed error O, is calculated. This comparison, of course, will be done only for those trees for which $(1–E) \times \delta L_{join} < \delta L_{split}$, and how many cases fulfil this condition in turn will depend on the value of E. In the implementation of TNT, O (with mean, range and standard deviation) can optionally be reported after a search using branch-swapping concludes, or accessed with the scripting language of TNT, to fine-tune the searches with scripts.

As O can be calculated at any stage of the search, this can also be used to automatically set the error margin to a value that is defined by the data at hand, instead of fixed prior to the search. In TNT, it is possible to set the final error to either a fixed prespecified value, or the value O for the search so far (whichever

is largest or smallest, as set by the user). The value O plus a specified number of standard deviations is used (of course, the distribution of the error does not need to be normal, but using the standard deviation is the simplest approach). To specify the final error margin as the observed value, the > and < symbols are used after the last error margin; for example, the largest of either the default setting (0.15) or O plus 2 standard deviations is specified as *lmark errmarg 5-15 > 2/15-66*. When doing multiple addition sequences, the value O is recalculated again for each replication, for the number of cases observed between the initial stages of branch-swapping (i.e. as soon as the Wagner tree for the replication is finished) and the proportion P of swapping completed to switch to the final error margin. This is generally a conservative choice, because examination of several data sets suggests that the error produced by using δL_{join} tends to be larger when the trees are farther from optimal, decreasing towards the end of the branch-swapping cycle. This difference seems logical, as different positions for the landmark points at each node are more likely to be of similar optimality when the trees fit the data more poorly, thus making the approximation more error prone.

Mixing standard and landmark characters

In the implementation of TNT, it is possible to combine standard (one-dimensional, discrete or continuous) characters with landmark data. By default, every configuration is multiplied by a factor such that the sum of the maximum distances between landmark points (i.e. akin to the "range" in an additive character) equals unity, for each configuration; this can be changed by the user.

Catalano et al. (2015) and Catalano and Torres (in press) showed that the quality of phylogenetic analyses (as measured by congruence with other sources of data) increases with the number of landmark configurations. That is, just as it has always been held that standard phylogenetic analyses should include as much evidence as possible, so should studies using landmarks.

During branch-swapping, the standard characters are always evaluated first, for every set of rearrangements, using all the shortcuts and algorithms already in use in TNT (i.e. with the methods of Goloboff, 1996, 1999, 2015). The increase in length in the standard characters may allow rejecting many rearrangements without the need to examine the landmark characters at all; thus, combined matrices can be analysed faster than matrices with only landmark data. Note that this will not be the case when doing a full optimization of the landmark characters—it is unlikely that many rearrangements can be rejected on the basis of the discrete characters alone, as the contribution to the score from each landmark configuration is initially unknown (zero) in that case, instead of the scores of the two subtrees (to which only the length increment resulting from joining the subtrees remains to be added).

For branch-swapping, the length increments for the standard and the landmark characters when joining the two subtrees are calculated separately; the calculations for standard characters are exact, so the error margin only applies to the landmark portion (δL_{join}) of the length increase when reuniting subtrees.

Incremental Fermat recalculations (IFRs)

The FPST calculations consider that the position of points A–D in Fig. 2 (and those beyond) are fixed. However, global optimality may require that the position of those points is changed. It is then possible to use IFRs, radiating from the branch xy in Fig. 2. For example, point A will be recalculated as the Fermat point between x and the two descendants of A; if the position so calculated for A is different from the original one, then the descendants of A (if internal nodes) are added to the list of nodes to reposition. Likewise, point C will be recalculated as the Fermat point between y, the left descendant of C, and the ancestor of C. This IFR process can be repeated, branching from the union point a specified maximum number of branches (if no limit is set, then this is the same as the phase of iterative improvement of Goloboff and Catalano, 2011: 45), a given number of cycles. In TNT, the default is branching up to five nodes away from union point, with up to two cycles of improvement (this can be changed with the *lmark refine* command, or from the menu option **Settings/Landmarks/BranchSwapping**). Given that the optimal positioning of all the landmark points can only be achieved by globally considering all the points, the iterative process (which considers only one internal node at a time) can get trapped in local optima (for examples and discussion, see Goloboff and Catalano, 2011), but the tree scores are usually improved relative to FPSTs.

By applying IFRs and improving the score calculated by the FPST method, it is often possible to make acceptable (for re-checking with a full optimization) trees that would otherwise be rejected. The IFR is not as costly as a full optimization (of which the initial grid part is the slowest), but it does require more calculations than the simple FPST. The IFR process is then applied only to those rearrangements that exceed the current bound by a small margin, rejecting those rearrangements for which $(1–2E) \times \delta L_{\text{join}} > \delta L_{\text{split}}$ (when $E = 0$, the IFR process is never used in TNT). Given that trees that are marginally worse than the

best trees found so far are examined in more detail with the IFR, then it is possible to use lower values of E without increasing much the risk of incorrectly rejecting some rearrangements that should be accepted, thus producing a net speed gain.

Wagner trees

As is well known, the local optima common during branch-swapping become even more acute in the case of Wagner trees. Wagner trees decide the position of each taxon to be added to the growing subtree on the basis of only part of the evidence—only the states of the taxa added so far are considered (Goloboff, 2014: 126–127, provides extreme examples of the possible consequences of this). Given this situation, the positions of each landmark point are determined, after addition of each taxon, with only an iterative recalculation of Fermat points for each internal node in the tree (i.e. avoiding the use of the time-consuming Sankoff grid). This may produce positions which are farther from optimal than the position that would be obtained if the tree is reoptimized in full. This full reoptimization after every taxon insertion can be done optionally in TNT. However, the influence that this has on the final result is minimal, and the resulting Wagner trees have no noticeable differences in score when only the iterative Fermat recalculations are done after addition of each taxon. Evidently, the imprecision that comes from considering only part of the terminal taxa is more important than the imprecision in determining point locations with a faster method. In the data sets we have examined (from 39 to 160 taxa) fully reoptimizing the tree after addition of each taxon makes calculation of Wagner trees 5–15 times slower. As the Wagner trees are normally intended only to provide an acceptable initial point for further branch-swapping during searches, the faster approach is preferred as the default.

Tree-collapsing

When zero-length branches are to be eliminated, those landmarks for which the ancestor and descendant points have the same position (after re-checking tree scores with a full optimization) are considered to not be supported by the landmark in question. The most likely situation where this may happen is when the triangle formed by the descendants and the ancestor of the node in question has an angle of 120° or more, in which case the middle node will have the same position as the point corresponding to that angle. However, recall that the optimization identifies unique (hopefully optimal) positions for each landmark point, not being able to identify ambiguity (Goloboff and Catalano, 2011: 47). Thus, while the branches eliminated by collapsing will normally be branches that are indeed unsupported, other unsupported branches in the tree may not be identified as such. A consequence of this is that the different collapsing "rules" based on optimization [e.g. those discussed by Swofford and Begle (1993) and Coddington and Scharff (1994)], while still applicable in theory to the problem, make no difference in practice—no branch can be supported "ambiguously" by a method of landmark optimization that cannot identify ambiguity. The tree that results from collapsing branches based on individual reconstructions will necessarily be of the same score as the original binary tree (any difference in score between the two trees will arise from a failure to find optimal landmark positions for one or both trees, which is possible given the heuristic nature of the algorithms for optimization). Of course, given that landmark coordinates are usually measured with some degree of precision, and all taxa included in an analysis have slightly different coordinates for all the landmark points, it is unlikely that many branches will appear as unsupported by all landmarks, so that collapsing zero-length branches for landmark data normally eliminates no (or very few) branches.

TNT also implements the option of collapsing trees based on branch-swapping (as in Goloboff and Farris, 2002), and this is implemented for landmark data in the same way as for standard characters: for any rearrangement that produces a tree of the same score (or within the score difference specified with the command *subopt* or the menu option **Analyze/Suboptimal**), all the nodes between the two positions of the moved clade are marked to be collapsed. If a relative fit difference is specified, this is optionally calculated for the individual landmarks or for whole configurations, as discussed below in the section on resampling and Bremer supports. To avoid problems arising from almost exact ties in optimality, TNT considers scores that are within 10^{-10} units to be identical (this minimum difference is also used for determining "equally" optimal trees).

Terminal points and taxon deactivation

The grid for the initial phase of the optimization can be enlarged by adding the positions of terminal points to the grid (see Goloboff and Catalano, 2011: 44); this is the default setting of TNT (set with the *lmark termpoints* option). In the implementation of TNT, when some taxa are deactivated, the grid will still contain the points for inactive taxa. This increases the chances of finding better locations for the points. A consequence of this is that physically eliminating

some taxa from the matrix (e.g. by editing the file and deleting them) is not equivalent to leaving them in the matrix and deactivating them—the physical elimination may produce slightly worse scores.

Resampling and relative Bremer supports

Each landmark configuration is considered, in TNT, to constitute a "character". However, the individual landmarks of a given configuration may have some "internal" conflict or incongruence.[1] This raises the question of what units to consider for the purpose of measuring group supports: the individual landmarks or the entire configurations. In TNT, these two possible ways to measure group supports for landmarks are determined with the *confsample* option of the *lmark* command (or with the menu options **Settings/Landmarks/GeneralOptions**).

In the case of resampling, either the entire configuration or the individual landmarks can be resampled—whether the resampling is done with standard bootstrapping (Felsenstein, 1985), independent-removal jacknifing (Farris et al., 1996), Poisson bootstrapping or symmetric resampling (Goloboff et al., 2003). Of course, when only one or a few landmark configurations are being considered in the analysis, conflict can be evaluated meaningfully only by considering the individual landmarks.

For searching the resampled data sets, it is advisable to relax the search parameters (as in Farris et al., 1996), especially for large data sets. For example, in the case of the *Prevosti_2012* data set (150 taxa; see Table 1 and Appendix 2), setting TBR to reconnect/reroot no more than six nodes away from original position, using an initial error margin of $E = 0$, and switching to $E = 0.05$ when $P = 0.50$ (with the commands *bbreak: limit 6; lmark errmarg 5|50*), the searches proceed about five times faster than with the default parameters (relaxing rigour, of course), thus making it possible to complete 100 pseudoreplicates in about 8 h.

The absolute (Bremer, 1994) and relative (Goloboff and Farris, 2002) Bremer supports are calculated for landmarks in exactly the same way as for standard characters. That is, either (i) the user must search for trees that are suboptimal by a prespecified difference in score, then summarize the results, or (ii) TBR-swapping can be used to record the minimum degree of suboptimality (absolute or relative) to lose each group in the tree, without the need to actually save multiple suboptimal trees. The relative Bremer supports are based on calculating relative fit differences (*RFD*) between the optimal and the suboptimal tree, defined as $RFD = (F–C)/F$. For standard characters, F is the sum of step differences between the two trees, for the characters that fit the optimal tree better, and C is the sum of step differences for the characters that fit the suboptimal tree better. In the case of landmarks, the quantities F and C can be calculated either with the differences of fit in individual landmarks, or in the entire configurations (as in resampling, calculating relative Bremer supports for whole configurations is meaningful only when several configurations are included in the matrix). When implied weighting is in effect, the quantities F and C are determined from differences in score instead of differences in raw numbers of steps (see below, in section on Implied Weights, for details on how the score is measured under extended implied weights).

Sectorial search, tree-fusing, ratchet and drifting

The previous sections discuss details of the implementation of branch-swapping and Wagner trees, which constitute the core for the search algorithms. The new version of TNT also implements the other search algorithms that were already available for standard characters.

Sectorial searches (Goloboff, 1999) for standard characters are based on creating new terminal taxa or hypothetical taxonomic units (HTUs), based on the state sets calculated during the down pass of the optimization. Given that this takes into account ambiguity, it has the great advantage that all the score differences between trees for the reduced data set must match exactly the score differences for the entire data set (see Goloboff and Pol, 2007); thus, finding a reduced tree which (for the reduced data set) is X steps shorter than the original resolution guarantees that grafting the new solution into the entire tree will be X steps shorter for the entire data set. In the case of (symmetric) Sankoff characters, the extra cost of every possible state (see Goloboff, 1998) during the down pass must be considered in addition to the optimal states. Given that the optimization for landmark data does not identify ambiguity or suboptimality, it is not possible to create HTUs such that the score differences between the reduced data set match those for the entire data set. The same is true of auto-weighted optimization (Goloboff, 1997) and asymmetric Sankoff characters. For those characters, as well as for landmark characters, the sectorial searches are implemented in TNT by using constraints, such that the sector chosen is left unconstrained, and a constraint of monophyly for all the other groups in the tree is in effect. This produces the same effect as a sectorial search, improving the tree one sector at a time. However, it is not as time-efficient

[1] Recall that two characters are said to be congruent when they can both be free of homoplasy on some tree(s), and that "homoplasy" for a landmark is defined as the observed length minus the minimum possible length on any tree.

Table 1
Analysis of RAS+TBR-swapping (command *mult = rep 1 hold 1*). All data sets were run with random seed 1 and default settings. Columns: taxa; number of landmark configurations; number of individual landmarks; maximum dimensions of landmarks; number of rearrangements examined to complete the search ("Rearrangs"); number of changes to the Wagner tree made during branch-swapping ("Substs"); total search time used (in seconds); proportion of the search time used in rechecking scores with a full optimization ("Recheck").

Dataset	Taxa	Confs	Lands	Dim	Rearrangs.	Substs.	Time	Recheck
Pierce_2008	24	1	65	2	18 160	28	25.13	0.26
Claverie_2013	25	3	88	2	21 943	18	39.00	0.21
Rios_landmarks	25	6	60	2	17 671	17	25.41	0.35
Foth_2012	31	1	21	2	33 282	30	18.03	0.36
Astua_2009	32	1	14	2	48 354	49	14.23	0.26
Outomuro_2013	32	2	23	2	33 886	27	18.66	0.34
Piras_2012	32	1	50	2	59 836	46	62.36	0.22
Abe_2012	34	1	19	2	68 096	43	28.00	0.21
Rios_combined	34	6(+89)	60	2	56 726	32	10.38	0.83
Alvarez_2013	39	4	90	2	69 823	33	141.53	0.20
Stubbs_2013	53	1	74	2	252 624	62	387.09	0.18
Franklin_2014	59	1	15	2	349 603	71	115.25	0.18
Montero	61	5	124	2	353 106	55	1319.06	0.31
Vera_Candiotti_2009	108	1	24	2	1 819 189	156	940.36	0.15
Prevosti_2012	151	1	29	2	6 632 950	243	4226.84	0.17
Klingenberg_2013	160	1	11	2	6 515 829	228	1715.94	0.21
GJ_GR	9	14	476	3	405	0	17.30	0.82
AP_APV_2013	24	2	56	3	12 033	24	198.63	0.93
Almecija_2015	26	1	23	3	15 661	21	109.31	0.92
Aristide_2013	29	1	35	3	21 796	30	135.16	0.91
Baab_2014	33	1	39	3	52 784	45	253.94	0.88
Martín-Serra_2014	46	6	104	3	174 607	36	1609.59	0.80

as the actual creation of reduced data sets used for standard characters (which allows much faster optimizations, by virtue of internally using reduced trees with fewer nodes and eliminating the characters that are not informative for the reduced data set).

Similar considerations apply to tree-fusing (Goloboff, 1999). In the case of standard characters, the scores resulting from mixing parts of two trees can be deduced quickly with incremental down-pass optimizations (as in Gladstein, 1997). That is not possible for landmark characters. TNT therefore uses the expediency, for landmark characters, of simply doing a full optimization for the mixed trees (for simplicity, when the data set contains some landmark characters, the full optimization is also used for the standard characters). Although tree-fusing often exchanges only a few subtrees and therefore does not take as much time as a random addition sequence plus branch-swapping, the increase in efficiency by using tree-fusing for landmark data will not be as dramatic as for standard characters. In addition, data sets with only one or a few landmark configurations and no standard characters tend to be poorly structured, so that the final results of searches using different starting points tend to be rather different, and not very useful for tree-fusing. This relative lack of structure suggests that the alternative algorithms for tree-hybridization described by Goloboff (2014) might perhaps be more useful for landmark data, but no strict comparisons of the effectiveness of those newer algorithms have been carried out yet.

In the case of ratchet (Nixon, 1999) and tree-drifting (Goloboff, 1999), the reweighting and the calculations of relative fit differences (respectively) to produce the perturbed search can be done either for the entire configurations or for the individual landmarks (see previous section). Given that landmark data tend to be poorly structured, using strong perturbation factors (e.g. a high probability of reweighting, in the case of ratchet, or a high probability of acceptance, in the case of drifting; accepting a large number of substitutions to the tree during the perturbation phase, in either case) tends to produce perturbed trees that are too far from optimal, so that the subsequent search phase can be almost as long as a search that starts from an RAS Wagner tree. Thus, it is advisable to set those factors to more conservative values in such a way that the trees resulting from the perturbation phase are only slightly less optimal than the previous trees (e.g. by increasing *xfactor* for the *drift* command, or increasing the "rejection factor" for the menu option **Analyze/NewTechSearch/DriftSettings**, and setting the percentage of swapping to the tree to interrupt the perturbation phase to a lower number).

Implied weighting, standard and extended

Implied weighting for landmarks is available in TNT, with the default weighting function, with different values of concavity k (see Goloboff, 1993). As in

the case of continuous/Sankoff characters, no user-defined weighting functions are available for landmark characters. Extended implied weighting (Goloboff, 2013) is also available; the only limitation is that sets for weighting based on average homoplasy must include either landmark or standard characters; no set can combine both types of characters.[2]

The method of implied weighting is based on considering the amounts of homoplasy for the different characters, and this makes it necessary to calculate (for every individual landmark) the best score that the landmark could have on any tree. This comes down to calculating a euclidean Steiner tree for the landmark, a problem long known (e.g. since Karp, 1972) to be complete.

To avoid that, for landmarks, the former implementation of implied weighting (i.e. prior to version 1.5) used as the minimum possible the sum S of distances between the three terminal taxa A, B, C, and their Fermat point, with the three terminal taxa chosen so that S is maximum; no tree including all taxa can have a length smaller than this value S. This value S, however, may be too small to be achievable by any single tree containing all the terminal taxa and will thus lead to overestimating the homoplasy (and not necessarily by the same factor in all landmarks). Given that the new version of TNT contains search algorithms, and that these seem rather effective in finding Steiner trees for a single landmark (a much easier task than finding the tree that minimizes the sum of displacements for all landmarks taken together, because for a single landmark there is no incongruence), the option to calculate the minimum for each landmark by searching was added to the program. This option is available with the *lmark usmin =!* command (or with **Settings/Landmarks/General Options**, in the Windows version), and the values produced can be saved to a file and subsequently read into the program (so that the minima do not need to be calculated again every time the data set is loaded in memory). To save the minima (once calculated) to a file called *minima.txt* that can be subsequently read by TNT you need to first open a log file, then save the minima (by calling *lmark usmin* with no further arguments) and then close the log file:

log minima.txt; lmark usmin; log/;

The minimum values calculated in this way will allow much more precise evaluations of homoplasy for the landmark characters.

It is possible to weight either using the homoplasy of individual landmarks or that of entire configurations (i.e. giving all landmarks in a configuration the same weight). In the current version of TNT (unlike the previous version), to weight one or more configurations by the total homoplasy of the component landmarks, a set of configurations for uniform weighting under extended implied weighting (Goloboff, 2013) must be defined, with the *xpiwe[* command, for each landmark character to be weighted with the total homoplasy in the configuration (note that this differs from the case of standard characters, for which defining one-character sets for weighting with average homoplasy produces no difference in results). When a group includes more than one landmark configuration, the average homoplasy for all the configurations is used.

When sets for uniform weighting have been defined, the relative Bremer supports are calculated by considering the relative contributions of each landmark or configuration to the total score of the (set of) configuration(s). This is the same way in which relative Bremer supports are treated for discrete characters, by considering the proportional amount of homoplasy of the character relative to the homoplasy of the entire set. Then, if the total homoplasy for the set is H and the score is S, the score contribution by an individual landmark will be $S \times h_{\text{land}}/H$ (where h_{land} is the homoplasy of the individual landmark). If the user chooses to consider differences in fit between entire configurations, the contribution of an individual configuration within the set will be instead $S \times h_{\text{config}}/H$ (obviously this makes a difference only for weighting sets comprising more than a single configuration). Either way, the differences in score contributions for the different landmarks/configurations are what is used to calculate the relative fit differences between the trees compared.

Another problem that must be considered for implied weighting is scaling: the amounts of homoplasy are calculated as differences in the original units of measurement. Therefore, changing the units of measurements may affect the results for implied weighting. This differs from the default treatment of landmarks under *prior* weights, which rescales each configuration to unity (i.e. using factors to multiply the differences such that the maximum pairwise distances between two taxa for all the landmarks in a configuration sums up to one). Mongiardino Koch et al. (2014) showed that, although the use of implied weighting decreases the effect of differences in scaling (as first proposed by Goloboff et al., 2006), the use of a uniform scaling is more effective at making the contributions of different characters more even. Thus, for weighting landmark data, it is advisable to use a uniform scaling of the landmarks (e.g. the command *lmark rescale =**; will automatically rescale all landmark configurations to unity; see *help lmark* for details or other options).

[2] This limitation arises from the need to keep length increments (during branch swapping) for standard and landmark characters separate, given that only the latter require re-checking with a full optimization, and these length increments would have to be tracked separately for each set of characters, for each rearrangement.

A further complication is that, when weighting landmarks individually, the score for each landmark is multiplied by the inverse of the number of landmarks in the configuration (see Goloboff and Catalano, 2011: 47). As a consequence, the expected results are not always easily intuited. Figure 3 shows two data sets, each with two conflicting configurations (separated by "|"). Real examples will of course always have configurations with three or more landmarks; the example has only one or two, to make it easier to visualize; this does not affect the way in which the score calculations are done. Also, it is unlikely that real examples will have different landmarks in a configuration sharing some identical coordinates; again, this does not affect the phylogenetic calculations, and is used for simplicity of visualization. In data set 1, the first configuration supports one of the alternative trees (tree 1) with each of its landmarks; in data set 2, only half the landmarks support tree 1. The second configuration of both data sets, instead, supports tree 2 with each of its landmarks. Weighting landmarks individually or weighting configurations, in the original scale or rescaling using the default landmark factors, leads to prefer either tree 1, 2 or both. The decision to weight landmarks or configurations may well depend on what the configuration represents, or how the landmarks were chosen—a data set may well comprise some configurations best weighted globally, and some for which the individual landmarks are weighted.

Empirical evaluation and results

Branch-swapping

The exact time gain resulting from application of the FPST/IFR shortcuts is hard to calculate. Given one landmark, the time needed to calculate the FPST for each rearrangement is roughly constant on the number of taxa; but a proportion of the rearrangements will appear to improve the score, requiring a full optimization of the resulting tree. The number of rearrangements that require a full optimization will vary with data sets, and with the error margin. Note that the number of rearrangements examined per second, or the number of rearrangements until a search (e.g. a Wagner tree plus TBR) is completed, may not be strictly comparable between searches using FPST/IFR

DATA SET 1

a	0,0	0,0	0,0
b	0,0	0,1	0,1
c	0,1	0,0	0,0
d	0,1	0,1	0,1

DATA SET 2

a	0,0	0,0	0,0	0,0
b	0,0	0,0	0,1	0,1
c	0,1	0,0	0,0	0,0
d	0,1	0,0	0,1	0,1

Data set	Weight,scale	Tree 1			Tree 2			Tree(s) chosen
		Conf 1	Conf 2	Tot	Conf 1	Conf 2	Tot	
1	land,orig	0.000	0.125 + 0.125	0.250	0.250	0.000 + 0.000	0.250	1 = 2
	conf,orig	0.000	0.200 + 0.200	0.400	0.250	0.000 + 0.000	0.250	2
	land,resc	0.000	0.071 + 0.071	0.143	0.250	0.000 + 0.000	0.250	1
	conf,resc	0.000	0.125 + 0.125	0.250	0.250	0.000 + 0.000	0.250	1 = 2
2	land,orig	0.000 + 0.000	0.125 + 0.125	0.250	0.125 + 0.000	0.000 + 0.000	0.125	2
	conf,orig	0.000 + 0.000	0.200 + 0.200	0.400	0.250 + 0.000	0.000 + 0.000	0.250	2
	land,resc	0.000 + 0.000	0.071 + 0.071	0.143	0.125 + 0.000	0.000 + 0.000	0.125	2
	conf,resc	0.000 + 0.000	0.125 + 0.125	0.250	0.250 + 0.000	0.000 + 0.000	0.250	1 = 2

Fig. 3. Example of behaviour under implied weighting for two data sets (1 and 2), when either individual landmarks or entire configurations are weighted (using rescaling with default factors, to make sums of maximum point distances for each configuration equivalent). The score of each landmark (and the total) is indicated for each tree. Depending on whether individual landmarks or configurations are weighted, and whether landmarks are rescaled or not, different trees are selected as optimal. The command to set weighting for entire configurations is *xpiwe=; xpiwe [0; xpiwe [1*; the weighting for individual landmarks (the default) is set with *xpiwe–;*. Rescaling with current factors (the default, if none defined) is achieved by typing *lmark rescale =*;* (this cannot be undone, except by re-reading the data set into memory).

and searches that use a full optimization for each rearrangement—some of the rearrangements rejected during the FPST/IFR search may actually be acceptable using a full optimization, thus biasing the comparison. And, when the error margin used varies during the search (as explained in the section on "Error margin"), the likelihood of a rearrangement being incorrectly rejected changes during the search (lowest at the end). The time gain, therefore, cannot be easily predicted from theory, but must instead be observed in practical examples. This is difficult, especially because searches when the shortcuts are not used take so much time as to make strict comparisons prohibitive. A variety of 2D and 3D landmark data sets, with numbers of taxa ranging from nine to 160 (average 48), and number of landmarks from 11 to 476 (average 68), was used to compare the speed of rearrangement evaluation during searches with the speed of full optimizations (see Appendix 2 for the sources of the data sets), to obtain an approximation to the expected speed ratios. Comparing only the speeds of rearrangement evaluation may slightly overestimate the actual ratio of the times needed for an entire search done with both approaches, but probably not by more than 10 or 15%. All the optimization and search parameters used are the default ones (i.e. grids of 6 × 6 in the case of 2D, and 6 × 6 × 6 in the case of 3D, nested once with a window of 1, adding terminal points to the grid; error margin starting at 0, switching to 0.05 and 0.15 when the proportion of swapping reaches 15 and 66%, respectively). The results are shown in Tables 1 and 2. For actual data sets, only a relatively low proportion of rearrangements require a full optimization, with clear differences for 2D and 3D cases in the proportion of search time used in rechecking scores with a full optimization. For 2D, from 0.15 to 0.36 of the search time (see Table 1) is used in rechecking scores; the exception is the combined data set (*Rios_combined*), for which the proportion of time used in rechecking is 0.83 (this is not because the rechecking uses more time, but instead because the shortcuts for quick evaluation during searches use much less, as explained under "Mixing standard and landmark characters"). In the case of 3D, the proportion of time used in rechecking scores is much larger, from 0.80 to 0.93 of the total search time; this is because the time for FPST/IFR remains more or less the same, but for the full optimization the time used in the initial calculation of the Sankoff grid is much more important. The proportion of moves examined during a search that require rechecking is rather low, with only small differences between 2D and 3D, and decreases with the number of taxa. Thus, in practice, the time saved by applying the FPST/IFR method increases with number of taxa faster than linearly (Fig. 4A,B). Although the speed ratio increases more or less

Table 2
Other results for the runs of Table 1. Columns: number of trees evaluated per second during the search ("Speed(search)"); observed error, calculated when rechecking scores with a full optimization ("Error"); number of rearrangements that had to be rechecked with a full optimization ("Cases"); number of rearrangements per second examined with a full optimization per rearrangement ("Speed(full)"); ratio of speeds of rearrangement evaluation between an FPST/IFR search and a search using a full optimization to evaluate each rearrangement ("Ratio"); ratio of speeds divided by the number of taxa ("Ratio/Taxa"); ratio between the number of rearrangements examined during the search and the number of rearrangements that had to be rechecked with a full optimization ("Rearr/Cases").

Dataset	Speed (search)	Error	Cases	Speed (full)	Ratio	Ratio/Taxa	Rearr/Cases
Pierce_2008	722.79	7.90 ± 0.70	171	30.77	23.49	0.98	106.20
Claverie_2013	562.64	6.07 ± 0.58	132	18.03	31.20	1.25	166.23
Rios_landmarks	695.54	6.71 ± 0.72	256	30.33	22.93	0.92	69.03
Foth_2012	1845.72	5.80 ± 0.50	379	62.15	29.70	0.96	87.82
Astua_2009	3397.08	9.21 ± 0.76	277	87.64	38.76	1.21	174.56
Outomuro_2013	1816.26	6.95 ± 0.32	344	58.21	31.20	0.98	98.51
Piras_2012	959.53	7.21 ± 0.45	298	25.20	38.07	1.19	200.79
Abe_2012	2432.00	5.88 ± 0.38	334	65.32	37.23	1.10	203.88
Rios_combined	5467.57	–7.29 ±3.01	133	19.28	283.66	8.34	426.51
Alvarez_2013	493.34	8.33 ± 0.41	265	10.53	46.87	1.20	263.48
Stubbs_2013	652.62	10.10 ± 0.47	494	8.08	80.76	1.52	511.38
Franklin_2014	3033.43	2.71 ± 0.62	590	31.53	96.22	1.63	592.55
Montero	267.69	3.22 ± 0.20	1508	3.76	71.15	1.17	234.16
Vera_Candiotti_2009	1934.57	8.77 ± 0.49	977	8.05	240.31	2.23	1862.02
Prevosti_2012	1569.24	7.57 ± 0.25	1988	3.12	503.63	3.34	3336.49
Klingenberg_2013	3797.24	4.81 ± 0.18	2530	7.53	504.35	3.15	2575.43
GJ_GR	23.41	7.02 ± 0.43	59	4.17	5.61	0.62	6.86
AP_APV_2013	60.58	7.28 ± 0.34	313	1.83	33.07	1.38	38.44
Almecija_2015	143.27	5.33 ± 0.27	414	4.34	33.00	1.27	37.83
Aristide_2013	161.27	7.50 ± 0.48	264	2.39	67.48	2.33	82.56
Baab_2014	207.86	6.46 ± 0.52	377	1.89	110.00	3.33	140.01
Martín-Serra_2014	108.48	4.37 ± 0.16	581	0.48	227.65	4.95	300.53

regularly with taxa, the best predictor for the speed ratio between FPST/IFR and a full optimization seems to be the proportion of moves examined during the search that need to be rechecked with a full optimization, which for the landmark-only data sets show an almost perfectly linear relationship (Fig. 4C,D) for both 2D ($R^2 = 0.964$ when the combined data set is excluded) and 3D ($R^2 = 0.999$). The difference in slope between 2D and 3D arises because of the differences in relative times needed for FPST/IFR and full optimization. In the end, the net result of applying the shortcut is that (for the cases with over 30 taxa), searches for T terminal taxa take between roughly T and $3T$ times faster than if using a full optimization, with one exception in the *Martin-Serra_2014* data set (where the search is almost $5T$ times faster than if using a full optimization). For the largest cases examined (151–160 taxa), searches with the FPST/IFR are expected to be about 500 times faster than searches using a full optimization. The case of the combined data set (*Rios_combined*) shows that, when combined with standard characters, the gain in time by applying the shortcuts described here is even more significant, making searches $8T$ times faster than if using a full optimization.

Comparison of other search algorithms

Wagner trees and branch swapping are only the basic search algorithms; Table 3 shows the results of running more elaborate search algorithms in six of the most difficult data sets. As discussed above, the implementation of those algorithms cannot match the efficiency attainable with standard characters, but the

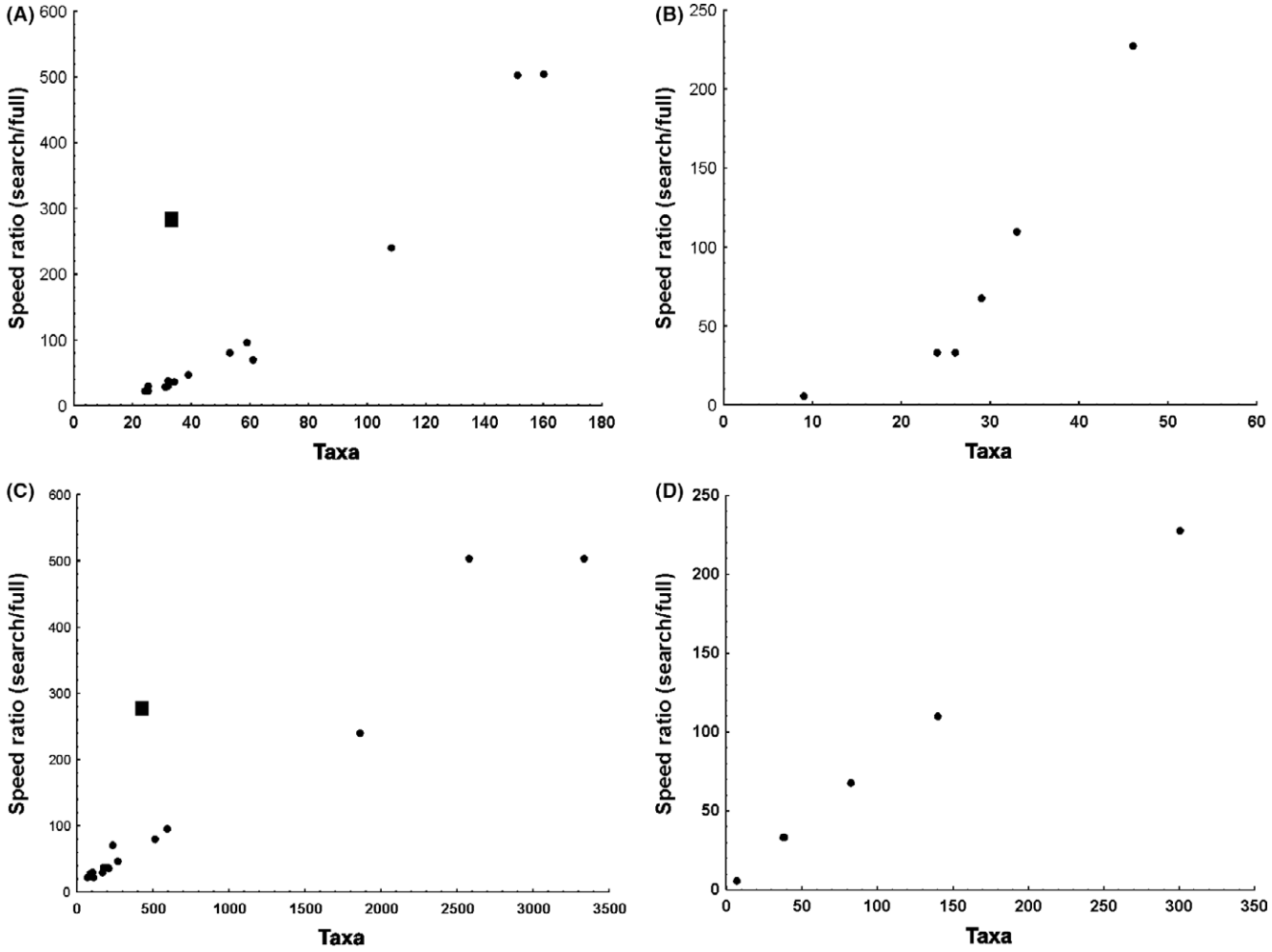

Fig. 4. Graphs showing the efficiency of the FPST/IFR shortcuts, for both 2D (left, A, C), and 3D (right, B, D). (A,B) Ratio between speed of rearrangement evaluation using the shortcuts and using a full optimization, as a function of the number of taxa. (C,D) Ratio between speed of rearrangement evaluation using the shortcuts and using a full optimization, as a function of the proportion of branch-swapping moves that need to be re-checked with a full optimization during swapping. For 2D data, the square indicates the data set combining landmark and standard characters. See text for additional discussion.

Table 3
Comparison between different search algorithms, for six representative data sets. Each of the routines was run 32 times, using the same initial random seeds for all routines. "Sect" is an RAS+TBR followed by a random sectorial search (RSS, dividing in pieces of 8–15 nodes, or 12–25, default parameters otherwise). "Drift" is five rounds of the *drift* command (*drift = iter 5*), with otherwise default parameters. "Xmult" is the *xmult* command with default parameters (five replications of RAS+TBR+RSS+tree-fusing at the end). The first line of score indicates the best score in the 32 replications for the routine (if followed by a number *xN*, this indicates the number of times the routine found the best known score for the data set). For the "Sect" columns, a "+" symbol followed by a number indicates how many times the routine produced an improvement over the results of plain RAS+TBR. The second line indicates the averages, for both scores and times, and the third, the standard deviation. The times for "Sect" also include the time needed for RAS+TBR.

	RAS+TBR		SECT 8-15		SECT 12-25		DRIFT		XMULT	
	Score	Time	Score	Time	Score	Time	Score	Time	Score	Time
Alvarez_2013	23.8505x4		23.8505×4 + 2		23.8505×4 + 6		23.8505×4		23.8505×18	
	23.8662	116	23.8660	255	23.8645	308	23.8662	405	23.8545	1036
	±0.0083	±25	±0.0082	±16	±0.0066	±20	±0.0083	±31	±0.0047	±84
Claverie_2013	9.1425×9		9.1425×10 + 1		9.1425×10 + 1		9.1425×9		9.1425×10	
	9.1430	22	9.1425	78	9.1425	52	9.1429	101	9.1425	226
	±0.0028	±4	±0.0000	±5	±0.0000	±5	±0.0028	±10	±0.0000	±14
Martin_Serra_2014	37.2014×4		37.2014×7 + 16		37.2014×8 + 15		37.2014×4 + 9		37.2014×10	
	37.2280	2274	37.2180	5755	37.2155	5757	37.2221	5304	37.2023	19060
	±0.0282	±621	±0.0269	±333	±0.0258	±379	±0.02136	±586	±0.0019	±2497
Franklin_2014	3.4991×13		3.4991×20 + 8		3.4991×19 + 9		3.4991×13 + 16		3.4991×30	
	3.5114	67	3.5085	189	3.5070	231	3.5010	301	3.4998	1123
	±0.0120	±10	±0.0126	±18	±0.0114	±25	±0.0041	±17	±0.0037	±67
Prevosti_2012	10.8933		10.8933 + 23		10.8927×1 + 29		10.8933 + 9		10.8933	
	10.9143	2780	10.9112	5768	10.9105	6646	10.9115	8696	10.9013	29462
	±0.0128	±462	±0.0115	±537	±0.0109	±665	±0.0102	±811	±0.0044	±3562
Klingenberg_2013	13.1498		13.1455 + 18		13.1455 + 18		13.1498 + 5		13.1371×1	
	13.1869	1140	13.1832	2330	13.1823	2590	13.1855	3211	13.1657	11545
	±0.0210	±230	±0.0196	±233	±0.0204	±293	±0.0195	±286	±0.0127	±1451

results are still clearly improved. Applying random sectorial searches (RSSs) to the results of RAS+TBR produced additional improvements in score, more often so in the largest data sets (*Klingenberg_2013* and *Prevosti_2012*, in 56–90% of the runs). Adding a sectorial search takes, on average, 60% longer than the RAS+TBR alone. The effectiveness of tree-drifting seems similar to that of sectorial search in medium-sized data sets, but inferior in the two largest data sets (producing slightly worse average scores; see Table 3). The *xmult* command runs (by default) five replications of RAS+TBR+RSS, and then submits those results to tree-fusing. For all but the easiest data sets (*Alvarez_2013* and *Claverie_2013*), the average scores produced by *xmult* are better than the average scores by any of the other routines, and it produces a larger number of the hits to the best known scores for each data set (except for *Prevosti_2012*). The *xmult* command takes about ten times longer than a single plain RAS+TBR, but it still runs in a reasonable amount of time (3–8 h, for the data sets with 160 and 151 taxa, respectively—the latter has fewer taxa, but has almost three times as many landmarks).

Conclusion

The possibility of automatically importing data files in the TPS format into TNT greatly facilitates phylogenetic analyses of landmark data. The new search algorithms incorporated into TNT are effective for the analysis of landmark data sets, both in isolation or combined with standard characters. These algorithms make phylogenetic analyses with landmarks possible on standard personal computers, reliably finding optimal or near-optimal trees even for relatively large-sized landmark data sets.

Acknowledgments

We appreciate the comments and encouragement from many colleagues, including Salvador Arias, James Carpenter, James Farris, Marcos Mirande, Kevin Nixon, Martín Ramírez and Claudia Szumik. Support from CONICET (PIP 00687 to P.A.G., PIP 0260 to S.A.C.), the ANPCyT (PICT1679, PICT 1930 to S.A.C.) 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, with participation from P.A.G.) was also deeply appreciated. The users of TNT who have reported bugs and suggested improvements over the years are also thanked for their cooperation. Many of the data sets used for testing could be obtained thanks to the kindness of their authors. Finally, while this paper would have come out anyway, we found a lot more joy in writing it with the (negative) inspiration from the famous "*Malleus Cladisticarum, et Venaticus Veneficae*" (1487), by Jonathan Eisen-Kramer and other men of faith.

References

Brazil, M., Graham, R.L., Thomas, D. A., Zachariasen, M., 2014. On the history of the Euclidean Steiner tree problem. Arch. Hist. Exact Sci. 68, 327–354.

Bremer, K., 1994. Branch support and tree stability. Cladistics 10, 295–304.

Catalano, S., Goloboff, P., 2012. Simultaneously mapping and superimposing landmark configurations with parsimony as optimality criterion. Syst. Biol. 61, 392–400.

Catalano, S.A., Goloboff, P., Giannini, N., 2009. The optimization of landmark data: a three-dimensional approach. In: Szumik, C. and Goloboff, P. (Eds.), A summit of cladistics: abstracts of the 27th Annual Meeting of the Willi Hennig Society and VIII Reunión Argentina de Cladística y Biogeografía. Cladistics 26, 202–226.

Catalano, S.A., Goloboff, P., Giannini, N., 2010. Phylogenetic morphometrics (I): the use of landmark data in a phylogenetic framework. Cladistics 26, 539–549.

Catalano, S., Ercoli, M., Prevosti, F., 2015. The more, the better: the use of multiple landmark configurations to solve the phylogenetic relationships in musteloids. Syst. Biol. 64, 294–306.

Catalano, S., and Torres, A. 2016. Phylogenetic inference based on landmark data in 41 empirical datasets. Zoologica Scripta (in press).

Clemente, J., Ikeo, K., Valiente, G., Gojobori, T., 2009. Optimized ancestral state reconstruction using Sankoff parsimony. BMC Bioinformatics 10, 51–60.

Coddington, J., Scharff, N., 1994. Problems with zero-length branches. Cladistics 10, 415–423.

Farris, J. S., 1970. Methods for computing Wagner trees. Syst. Zool. 34, 21–24.

Farris, J. S. 1983. The logical basis of phylogenetic analysis. In: Platnick, N.I., Funk, V.A. (Eds.), Advances in Cladistics II. Columbia University Press, New York, pp. 7–36.

Farris, J. S., 2008. Parsimony and explanatory power. Cladistics 24, 825–847.

Farris, J., Albert, V., Källersjö, M., Lipscomb, D., Kluge, A., 1996. Parsimony jackknifing outperforms neighbor-joining. Cladistics 12, 99–124.

Felsenstein, J., 1985. Confidence limits on phylogenies: an approach using the bootstrap. Evolution 39, 783–791.

Giribet, G., 2005. A review of "TNT: Tree Analysis Using New Technology". Syst. Biol. 54, 176–178.

Gladstein, D., 1997. Efficient incremental character optimization. Cladistics 13, 21–26.

Goloboff, P., 1993. Estimating character weights during tree search. Cladistics 9, 83–91.

Goloboff, P., 1996. Methods for faster parsimony analysis. Cladistics 12, 199–220.

Goloboff, P., 1997. Self-weighted optimization: tree searches and character state reconstructions under implied transformation costs. Cladistics 13, 225–245.

Goloboff, P., 1998. Tree-searches under Sankoff parsimony. Cladistics 14, 229–237.

Goloboff, P., 1999. Analyzing large data sets in reasonable times: solutions for composite optima. Cladistics 15, 415–428.

Goloboff, P., 2013. Extended implied weighting. Cladistics 30, 260–272.

Goloboff, P., 2014. Hide and vanish: data sets where the most parsimonious tree is known but hard to find, and their implications for tree search methods. Mol. Phylogenet. Evol. 79, 118–131.

Goloboff, P., 2015. Computer science and parsimony: a reappraisal, with discussion of methods for poorly structured data sets. Cladistics 31, 210–225.
Goloboff, P., Catalano, S.A., 2011. Phylogenetic morphometrics (II): algorithms for landmark optimization. Cladistics 27, 42–51.
Goloboff, P., Catalano, S.A., 2012. GB-to-TNT: facilitating creation of matrices from GenBank and diagnosis of results in TNT. Cladistics 28, 503–513.
Goloboff, P.A., Farris, J.S., 2002. Methods for quick consensus estimation. Cladistics 17, S26–S34.
Goloboff, P., Pol, D., 2007. On divide-and-conquer strategies for parsimony analysis of large data sets: rec-i-dcm3 vs. TNT. Syst. Biol. 56, 485–495.
Goloboff, P., Szumik, C., 2015. Identifying unstable taxa: efficient implementation of triplet-based measures of stability, and comparison with Phyutility and RogueNaRok. Mol. Phylogenet. Evol. 88, 93–104.
Goloboff, P., Farris, J., Källersjö, M., Oxelman, B., Ramírez, M., Szumik, C., 2003. Improvements to resampling measures of group support. Cladistics 19, 324–332.
Goloboff, P., Farris, J., Nixon, K. 2004. TNT. In: Stevenson, D., Abstracts of the 22nd Annual Meeting of the Willi Hennig Society. Cladistics 20, 76–100.
Goloboff, P., Mattoni, C., Quinteros, S., 2006. Continuous characters analyzed as such. Cladistics 22, 589–601.
Goloboff, P., Farris, J.S., Nixon, K., 2008. TNT, a free program for phylogenetic analysis. Cladistics 24, 774–786.
González-José, R., Escapa, I., Neves, W., Cúneo, R., Pucciarelli, N., 2008. Cladistic analysis of continuous modularized traits provides phylogenetic signals in Homo evolution. Nature 453, 775–778.
Hovenkamp, P., 2004. Review of T.N.T.—Tree Analysis Using New Technology, Version 1.0, by P. Goloboff, J. S. Farris, and K. Nixon. Cladistics 20, 378–383.
Karp, R. M. 1972. Reducibility among combinatorial problems. In: Miller, R.E., Thatcher, J.W. (Eds.), Complexity of Computer Computations. Plenum, New York, pp. 85–103.
Klingenberg, C.P., Gidaszewski, N.A., 2010. Testing and quantifying phylogenetic signals and homoplasy in morphometric data. Syst. Biol. 59, 245–261.
Larsen, R., 2008. L1 generalized Procrustes 2D shape alignment. J. Math. Imaging Vis. 31, 189–194.
Lockwood, C., Kimbel, W., Lynch, J., 2004. Morphometrics and hominoid phylogeny: support for a chimpanzee-human clade and differentiation among great ape subspecies. Proc. Natl Acad. Sci. USA 101, 4356–4360.
Martín-Serra, A., Figueirido, B., Palmqvist, P., 2014. A three-dimensional analysis of the morphological evolution and locomotor behaviour of the carnivoran hind limb. BMC Evol. Biol. 14, 129.
Meier, R., Ali, F., 2005. The newest kid on the parsimony block: TNT (Tree analysis using new technology). Syst. Entomol. 30, 179–182.
Mongiardino Koch, N., Soto, M. I., Ramírez, M. J., 2014. First phylogenetic analysis of the family Neriidae (Diptera), with a study on the issue of scaling continuous characters. Cladistics 31, 142–165.
Nixon, K.C., 1999. The parsimony ratchet a new method for rapid parsimony analysis. Cladistics 15, 407–414.
Perrard, A., López-Osorio, F., Carpenter, J., 2015. Phylogeny, landmark analysis and the use of wing venation to study the evolution of social wasps (Hymenoptera: Vespidae: Vespinae). Cladistics. doi: 10.1111/cla.12138.
Rohlf, F.J. 1990. Rotational fit (Procrustes) methods. In: Rohlf, F.J., Bookstein, F.L., editors. Proceedings of the Michigan morphometrics workshop. Museum of Zoology, University of Michigan, Ann Arbor (MI), pp. 227–236.
Rohlf, F.J. 2002. Geometric morphometrics and phylogeny. In: MacLeod, N., Forey, P.L., editors. Morphology, shape, and phylogeny. Taylor & Francis, London, pp. 175–193.
Rohlf, F., Slice, D., 1990. Extensions of the Procrustes method for the optimal superimposition of landmarks. Syst. Zool. 39, 40–59.
Ronquist, F., Teslenko, M., van der Mark, P., Ayres, D., Darling, A., Höhna, S., Larget, B., Liu, L., Suchard, M., Huelsenbeck, J., 2012. MrBayes 3.2: efficient Bayesian phylogenetic inference and model choice across a large model space. Syst. Biol. 61, 539–542.
Siegel, A., Benson, R., 1982. A robust comparison of biological shapes. Biometrics 38, 341–350.
Stamatakis, A., 2014. RAxML Version 8: a tool for phylogenetic analysis and post-analysis of large phylogenies. Bioinformatics 30, 1312–1313.
Swofford, D. L., Begle, D. P. 1993. PAUP: phylogenetic analysis using parsimony. Version 3.1. Smithsonian Institution, Laboratory of Molecular Systematics, Washington, DC.
Swofford, D., Olsen, G., Waddell, P., Hillis, D. 1996. Phylogenetic inference. In: Hillis, D., Moritz, C., Mable, B. (Eds.), Molecular systematics, 2nd edn. Sinauer Associates, Sunderland, MA.
Wheeler, W. C., 1996. Optimization Alignment: the end of multiple sequence alignment in phylogenetics? Cladistics, 12, 1–9.
Wheeler, W. C., Lucaroni, N., Hong, L., Crowley, L. M., Varón, A., 2015. POY version 5: phylogenetic analysis using dynamic homologies under multiple optimality criteria. Cladistics 31, 189–196.

Appendix 1

Re-checking of tree-scores

Given that the length calculations during the search are approximate, it is necessary to recalculate tree scores in full before storing trees in the memory buffer. A problem that arises in this case is that the algorithms for length calculation are order-dependent, with regard both to how ambiguous optimizations are resolved during the initial grid estimation, and to the sequence with which the iterative Fermat calculations are done (see Goloboff and Catalano, 2011: 47). First, when the initial grid estimation for point locations produces two or more equally optimal locations, one is chosen at random. For each replication, searches may use different random seeds for producing lists of random sequences. Thus, a separate random number generator was added to TNT, such that the landmark optimization uses a random number generator with the same seed throughout the search. In addition, the random number generator for choosing ambiguous point locations is re-initialized for each landmark—otherwise, the score for one individual landmark will depend on which landmarks were optimized before. A more serious complication for obtaining identical scores is that during searches TNT, because it allows the user to select different taxon sets or outgroups, internally creates reduced trees and renumbers taxa so that the outgroup is always taxon 0. An example is shown in Fig. 5, which shows the node numberings for a tree with only seven out of ten taxa active during normal operation of the program (e.g. for character mapping, Fig. 5A), and the internal representation that is used by TNT during a search when terminal taxon 2 is chosen as the outgroup (numbering in Fig. 5B). Terminal taxa are renumbered, for the search, between 0 and the number of active taxa, A, minus 1 (0–6, in the example); the root node receives a number A, and the remaining internal nodes are numbered between $A+1$ and $2A$–2 (8–12, in the example). With this renumbering, listing the terminal taxa can go between 0 and A–1, without the need to consider whether a given terminal taxon (e.g. the terminal taxon originally numbered as 4) is included in the tree or not; additionally, taxon 0 is always sister to all the rest of taxa (also facilitating some operations to the tree). This renumbering is important for standard characters, for which length calculations are very fast, so that the manipulations to the trees themselves may proportionally occupy a significant amount of time (especially for very large numbers of taxa). The renumbering is perhaps not especially significant for landmark searches (where length calculations are very time consuming relative to the time required for tree

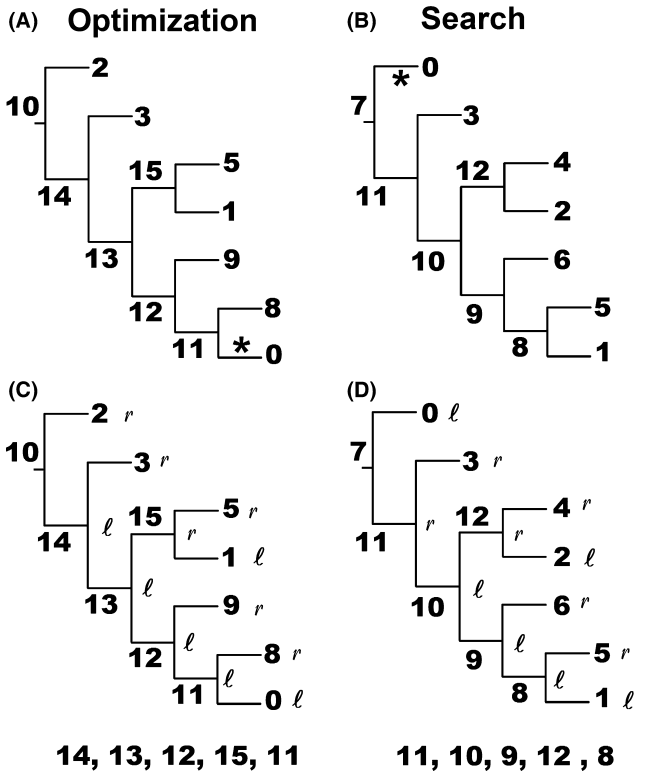

Fig. 5. Comparison between the numbering systems used during normal operations (e.g. optimization, score calculations, ancestral reconstructions) and during tree-searches. See text for discussion.

manipulations), but since TNT allows combining discrete and landmark characters, and all the code was already structured around this approach, it seemed best to preserve the existing implementation, reuse as much existing code as possible, and let landmark searches handle possibly reduced trees, with nodes renumbered.

To obtain identical scores for both numberings systems, TNT effects the iterative Fermat calculations following the sequence of a list for tree-traversing. For many trees, however, lists for tree-traversing are not unique; in Fig. 5C, for example, the lists 11, 10, 12, 9, 8 and 11, 10, 9, 8, 12 are (in addition to the list shown in the figure) equally valid lists for an up-pass. In the implementation of TNT, the algorithms to make lists for tree-traversing (down- and up-pass) guarantee that the resulting lists always place equivalent nodes at the same position in tree-traversing lists, regardless of node renumbering during searches. Trees are internally stored by TNT as ancestor-lists, where the ancestor anc_i for each node i is specified. To facilitate searches and optimizations TNT also uses lists of descendants; in the case of searches, only binary trees are relevant, and these require specifying only a left and right descendant for each node, as two separate lists, ℓ and r. Thus, in Fig. 5C, $\ell_{13} = 12$, and $r_{13} = 15$. The arrays ℓ and r are filled in such a way that a left/right correspondence is obtained from the tree numbering for both optimizations and searches. The algorithm to obtain this equivalence is as follows:

1) initialize a counter n_i of the number of descendants assigned to each internal node i to 0.

2) set p to the first (or the next) terminal taxon, starting from the lowest-numbered taxon. Set $i = anc_p$. Go to 3.

3) If $n_i = 0$, then set $\ell_i = p$, and $n_i = 1$. Otherwise (if $n_i = 1$), set $r_i = p$ and $n_i = 2$. Go to 4.

4) If $n_i = 1$ and $i \neq root$, set $p = i$, $i = anc_i$, and go to 3. Otherwise (if $n_i = 2$ or $i = root$), go to 2.

Given that only one terminal taxon can be designated as outgroup, there can be no more than one switch in taxon sequence in the reduced numbering system used during searches. Given that this switch will always involve the first split of the tree, the only node for which the left and right descendants may be switched for the reduced trees is the root of the tree (which always gets the same position in any tree-traversing list anyway); all the other nodes will necessarily have equivalent assignments of left and right descendants. Note that this way to obtain lists of left and right descendants will produce the same result regardless of the actual numbering of the internal nodes of the tree; switching the numbering for any two internal nodes in the trees does not alter the result (i.e. the result only depends on the sequence of the terminal taxa). This is advantageous for the tree searches, because it avoids the need to use a system for numbering internal nodes in such a way that equivalent nodes have the same number in identical trees (that system may be needed at some later point of the search, to compare whether tree topologies are equivalent, but is not needed for only rechecking tree score).

Once left and right descendants of each internal node have been assigned, the list L for node-traversing is filled with the following algorithm:

1) initialize $L_0 = root$, number of filled cells in the list to $f = 1$, and number of visited nodes to $v = 0$. Go to 2.

2) set $i = L_v$, and then $v = v + 1$. Go to 3.

3) if ℓ_i is an internal node, then set $L_f = \ell_i$, and $f = f+1$. If r_i is an internal node, then set $L_f = r_i$, and $f = f+1$. If $v < f$, then go back to 2. If $v = f$, stop.

After this process, the list L will be filled with f values, corresponding to all the internal nodes (the first one always being the root). Enumerating from L_0 to L_{f-1} produces an up-pass, and enumerating from L_{f-1} to L_0 a down-pass. In the algorithm, given that the left descendant is always added to the list of nodes to traverse before the right descendant (and given the equivalence in left and right descendants that results from the previous algorithm), the list to visit the nodes in an up-pass will always have equivalent nodes in exactly the same sequence, as shown in Fig. 5C and D. This tree-traversing list L is used to determine the sequence with which Fermat points are recalculated for the internal nodes of the tree, during both searches and normal operation of the program. This guarantees that recalculating tree scores will produce the same scores as had been calculated during the search (unless of course the random seed, or the parameters for the heuristic estimation of point locations, are changed by the user). The only aspect in which the calculation of tree scores is not replicated exactly during the search is when adding the lengths of each tree branch. The lengths of the branches themselves are calculated in the same way, but the addition to produce the final score may use a different ordering during the search (e.g. starting from the branch marked with an asterisk in Fig. 5A and B), which might conceivably produce (given that floating point operations are not necessarily exact) slightly different final numbers. These differences should be several orders of magnitude smaller than the differences that could arise from the factors discussed above, not noticeable to users.

Appendix 2

Source for the data sets used in the empirical comparisons

Abe_2012. – Abe, F. R. & Lieberman, B. S. (2012). Quantifying morphological change during an evolutionary radiation of Devonian trilobites. Paleobiology, 38, 292-307. Data from Dryad Digital Repository. http://dx.doi.org/10.5061/dryad.40r16 m35.

Almecija_2015. – Almécija, S., Orr, C. M., Tocheri, M. W., Patel, B. A. & Jungers, W. L. (2015). Exploring phylogenetic and func-

tional signals in complex morphologies: the hamate of extantanthropoids as a test-case study. The Anatomical Record, 298, 212–229.

Alvarez_2013. – Three data sets combined:

Álvarez, A. & Perez, S. I. (2013). Two- versus three-dimensional morphometric approaches in macroevolution: insight from the mandible of caviomorph rodents. Evolutionary Biology, 40, 150–157.

Álvarez, A., Ercoli, M. D. & Prevosti, F. J. (2013a). Locomotion in some small to medium-sized mammals: a geometric morphometric analysis of the penultimate lumbar vertebra, pelvis and hindlimbs. Zoology, 116, 356–371.

Álvarez, A., Perez, S. I. & Verzi, D. H. (2013b). Ecological and phylogenetic dimensions of cranial shape diversification in South American caviomorph rodents (Rodentia: Hystricomorpha). Biological Journal of the Linnean Society, 110, 898–913. Data from Dryad Digital Repository. http://dx.doi.org/10.5061/dryad.qg67c.

Aristide_2013. – Aristide, L., Rosenberger, A. L., Tejedor, M. F. & Perez, S. I. (2013). Modeling lineage and phenotypic diversification in the New World monkey (Platyrrhini, Primates) radiation. Molecular Phylogenetics and Evolution, 82, 375–385.

Astua_2009. – Astúa, D. (2009). Evolution of scapula size and shape in didelphid marsupials (Didelphimorphia: Didelphidae). Evolution, 63, 2438–2456.

Baab_2014. – Baab, K. L., Perry, J. M. G., Rohlf, F. J. & Jungers, W. L. (2014). Phylogenetic, ecological, and allometric correlates of cranial shape in Malagasy lemuriforms. Evolution, 68, 1450–1468.

Claverie_2013. – Claverie, T. & Patek, S. N. (2013). Modularity and rates of evolutionary change in a power-amplified prey capture system. Evolution, 67, 3191–3207. Data from Dryad Digital Repository. http://dx.doi.org/10.5061/dryad.67p55.

Foth_2012. – Foth, C., Brusatte, S. L. & Butler, R. J. (2012). Do different disparity proxies converge on a common signal? Insights from the cranial morphometrics and evolutionary history of Pterosauria (Diapsida: Archosauria). Journal of Evolutionary Biology, 25, 904–915. Data from Dryad Digital Repository. http://dx.doi.org/10.5061/dryad.4nr4cf37.

Franklin_2014. – Franklin, O., Palmer, C. & Dyke, D. (2014). Pectoral fin morphology of Batoid fishes (Chondrichthyes: Batoidea): explaining phylogenetic variation with geometric morphometrics. Journal of Morphology, 275, 1173–1186.

GJ_GR. – The composite of two data sets:

Gómez-Robles, A., Bermúdez de Castro, J. M., Arsuaga, J. L., Carbonell, E. & Polly, P. D. (2013). No known hominin species matches the expected dental morphology of the last common ancestor of Neanderthals and modern humans. Proceedings of the National Academy of Sciences of the United States of America, 110, 18196–18201.

González-José, R., Escapa, I., Neves, W. A., Cúneo, R. & Pucciarelli, H. M. (2008). Cladistic analysis of continuous modularized traits provides phylogenetic signals in Homo evolution. Nature, 453, 775–778.

Klingenberg_2013. – Klingenberg, C. P. & Marugán-Lobón, J. (2013). Evolutionary covariation in geometric morphometric data: analyzing integration, modularity, and allometry in a phylogenetic context. Systematic Biology, 62, 591–610. Data from Dryad Digital Repository. http://dx.doi.org/10.5061/dryad.787c0.

Martin-Serra_2014. – Martín-Serra A, Figueirido B, Pérez-Claros JA, Palmqvist P (2015) Patterns of morphological integration in the appendicular skeleton of mammalian carnivores. Evolution 69(2): 321–340. Data from Dryad Digital Repository. http://dx.doi.org/10.5061/dryad.m8440.

Montero. – An unpublished data set for Amphisbaenid genera, taken from skull photographs by R. Montero and J. Daza (pers. comm.).

Outomuro_2013. – Outomuro D., Dijkstra K. B., Johansson F. (2013) Habitat variation and wing coloration affects wing shape evolution in dragonflies. Journal of Evolutionary Biology 26, 1866–1874. Data from Dryad Digital Repository. http://dx.doi.org/10.5061/dryad.nh618.

Pierce_2008. – Pierce, S. E., Angielczyk, K. D. & Rayfield, E. J. (2008). Patterns of morphospace occupation and mechanical performance in extant crocodilian skulls: a combined geometric morphometric and finite element modeling approach. Journal of Morphology, 269, 840–864.

Piras_2012. – Piras, P., Sansalone, G., Teresi, L., Kotsakis, T., Colangelo, P. & Loy, A. (2012). Testing convergent and parallel adaptations in talpids humeral mechanical performance by means of geometric morphometrics and finite element analysis. Journal of Morphology, 273, 696–711.

Prevosti_2012. – Prevosti, F. J., Turazzini, G. F., Ercoli, M. D. & Hingst-Zaher, E. (2012). Mandible shape in marsupial and placental carnivorous mammals: a morphological comparative study using geometric morphometrics. Biological Journal of the Linnean Society, 164, 836–855.

Rios_combined and Rios_landmarks. – A matrix for *Actinopus* spiders (Araneae, Actinopodidae), from Ríos Tamayo, D. 2015. Estudios de morfología comparada en un marco filogenético, en arañas del género *Actinopus* (Mygalomorphae, Actinopodidae). Tesis Doctoral, presented to Universidad Nacional de Tucumán, 2015. The two versions are with landmarks only, and including 89 standard characters.

Stubbs_2013. – Stubbs, T. L., Pierce, S. E., Rayfield, E. J. & Anderson, P. S. L. (2013). Morphological and biomechanical disparity of crocodile-line archosaurs following the end-Triassic extinction. Proceedings of the Royal Society B, 280, 1–10. Data from Dryad Digital Repository http://dx.doi.org/10.5061/dryad.61s1n.

Vera_Candiotti_2009. – Vera-Candioti, M. F. & Altig, R. (2010). A survey of shape variation in keratinized labial teeth of anuran larvae as related to phylogeny and ecology. Biological Journal of the Linnean Society, 101, 609–625.