J Mol Evol (2006) 62:785–792
DOI: 10.1007/s00239-005-0176-2

JOURNAL OF MOLECULAR EVOLUTION

Relaxed Neighbor Joining: A Fast Distance-Based Phylogenetic Tree Construction Method

Jason Evans, Luke Sheneman, James Foster

Department of Biological Sciences, University of Idaho, P.O. Box 443051, Moscow, ID 83844-3051, USA

Received: 13 July 2005 / Accepted: 10 February 2006 [*Reviewing Editor:* Dr. James Bull]

Abstract. Our ability to construct very large phylogenetic trees is becoming more important as vast amounts of sequence data are becoming readily available. Neighbor joining (NJ) is a widely used distance-based phylogenetic tree construction method that has historically been considered fast, but it is prohibitively slow for building trees from increasingly large datasets. We developed a fast variant of NJ called relaxed neighbor joining (RNJ) and performed experiments to measure the speed improvement over NJ. Since repeated runs of the RNJ algorithm generate a superset of the trees that repeated NJ runs generate, we also assessed tree quality. RNJ is dramatically faster than NJ, and the quality of resulting trees is very similar for the two algorithms. The results indicate that RNJ is a reasonable alternative to NJ and that it is especially well suited for uses that involve large numbers of taxa or highly repetitive procedures such as bootstrapping.

Key words: Phylogenetic tree construction — Neighbor joining — Distance method

Introduction

The relaxed neighbor joining (RNJ) algorithm is a distance-based phylogenetic tree construction method that is similar to the neighbor joining (NJ) algorithm (Saitou and Nei 1987; Studier and Keppler 1988). RNJ and NJ have the desirable property that they are consistent estimators of phylogeny if the distances in a dataset are purely additive (Waterman et al. 1977). An additive dataset is one for which there exists a tree with nonnegative branch lengths that perfectly represents all of the pairwise distances. In practice, datasets are rarely additive, but NJ still works well as long as distances are nearly additive (Saitou and Imanishi 1989; Kuhner and Felsenstein 1994). NJ is one of the more commonly used tree construction methods, and in most cases it generates useful results.

NJ tree construction requires pairwise distances as input. Users of NJ typically start by calculating the magnitudes of differences between DNA sequences for the taxa under consideration. Those magnitudes are then treated as distances, and the NJ algorithm attempts to construct a tree that encodes all of the pairwise distances. The quality of results depends heavily on the accuracy of the distances, and several researchers have addressed this issue.

At the most basic level, distances can be corrected to account for multiple mutations at the same DNA site (Kimura 1980). Felsenstein (2004) uses a likelihood-based model for calculating distances. The Weighbor variant of the NJ algorithm reweights distances in an attempt to improve results when distantly related taxa are included in the dataset (Bruno et al. 2000). The BIONJ variant of NJ takes into account the variances and covariances of the distances and minimizes these at each step during tree construction (Gascuel 1997).

NJ is generally regarded as a fast reconstruction method, but its runtime complexity is $O(n^3)$, which

Correspondence to: Jason Evans; *email:* jevans@uidaho.edu

means that for large datasets, reconstruction is impractical. Mailund and Pedersen (2004) have developed QuickJoin, which implements a heuristic method that avoids considering pairs of nodes when they are known to fall outside bounds that are calculated from previous passes through the matrix of pairwise distances. Runtime results are good for this heuristic, as is shown later, but substantial extra space is required for auxiliary data structures, which limits QuickJoin to approximately 8000 taxa on modern 32-bit computer systems. As such, this heuristic approach is only compelling for a limited range of input sizes. Typical NJ implementations do not make use of such heuristics. Their runtimes are proportional to n^3 for all inputs, and they require space proportional to n^2. RNJ typically requires time proportional to $n^2 \lg n$, without using any more space than NJ.

Unlike simpler distance-based tree construction methods, NJ is able to deal with varying rates of evolution, so the resulting trees need not be ultrametric. This is accomplished by making join decisions based on transformed distances that take all taxa into consideration. A transformed distance T_{AB} between taxon A and taxon B is calculated as

$$T_{AB} = D_{AB} - \frac{\sum_{i=1, i \neq B}^{n} D_{Ai} + \sum_{i=1, i \neq A}^{n} D_{Bi}}{n - 2}$$

where D_{xy} is the distance between taxon x and taxon y, and n is the total number of taxa. The fractional sums represent the average distances from A and B to all other taxa. Note that the divisor is $n - 2$ because D_{AA} and D_{BB} are always zero, and D_{AB} is excluded, which means that two distances are effectively excluded from each sum.

There are two components that contribute to a minimal transformed distance. A small absolute distance between A and B contributes, but it is also important that, on average, A and B are farther from other taxa than they are from each other. By joining two taxa with globally minimal transformed distance between them at each step of tree construction, NJ builds the tree starting from the leaf nodes, without risk of joining taxa that are not immediate neighbors. Fig. 1 helps to illustrate why this is so. D_{CD} is the minimum pairwise distance, but T_{AB} and T_{EF} are the minimal transformed distances. This is because C and D are closer to the center of the tree than are the taxa with minimal transformed pairwise distances.

We modified the NJ algorithm in order to improve tree construction speed, with the additional goal of maintaining the quality of generated trees. This paper describes the algorithmic modifications that are the basis of RNJ, then presents experiments that measured tree construction speed and tree quality. The experimental results indicate that the RNJ algorithm is very fast, and that RNJ trees are of very similar quality to NJ trees.

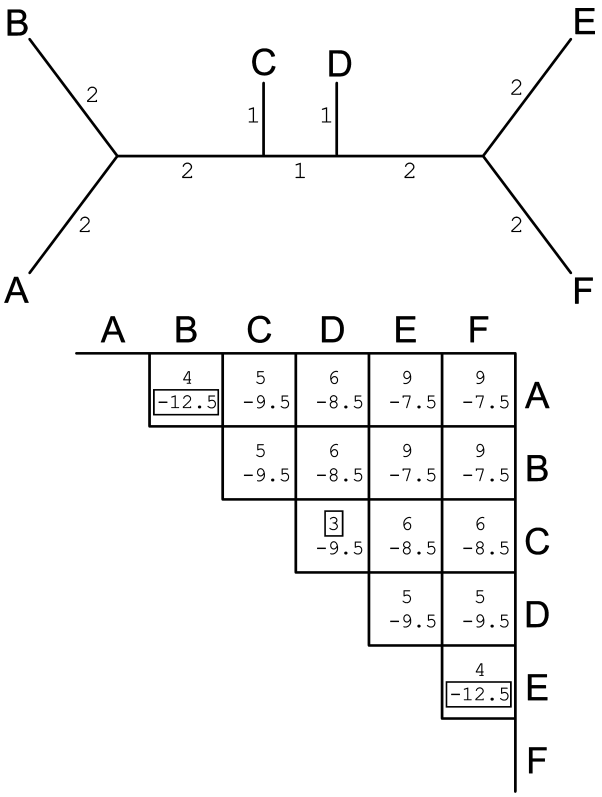

Fig. 1. Phylogenetic tree and corresponding patristic distance matrix. Each matrix cell contains the absolute pairwise distance (top) and transformed distance (bottom). D_{CD} is the minimum pairwise distance for this tree, but the associated transformed distance T_{CD} is not minimal; T_{AB} and T_{EF} are minimal. The NJ algorithm will first join A and B or E and F.

Algorithm

Like NJ, RNJ uses transformed distances when making join decisions, but rather than looking for a minimum among all transformed distances, RNJ looks for two taxa that have minimal transformed distance between them as compared to their transformed distances to all other taxa. Whereas NJ only joins pairs of neighboring leaf nodes that are minimally distant, RNJ can join any pair of plausible neighboring leaf nodes. There are typically many more pairs of neighboring leaf nodes than there are neighboring leaf nodes with minimal transformed distance, and RNJ is usually able to find such pairs without having to calculate transformed distances for all taxon pairs. We refer to the algorithm as "relaxed" NJ because it does not search for the globally minimal transformed distance; RNJ employs a less stringent, relaxed join criterion. Following is a general description of the RNJ algorithm, which operates on an input matrix of pairwise distances:

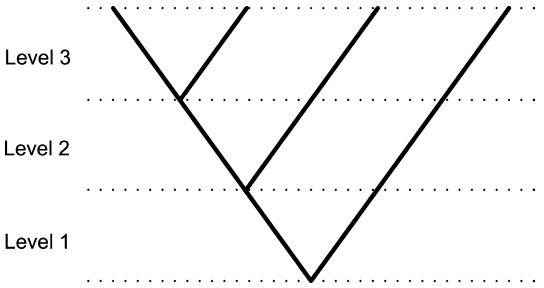

Fig. 2. A pectinate tree with $x = 4$ leaf taxa. The tree has $x - 1 = 3$ levels, and branches span from 1 to $x - 1 = 3$ levels.

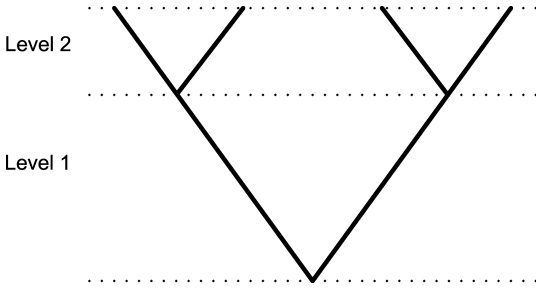

Fig. 3. A perfect tree with $2^{x=2}$ leaf taxa. The tree has $x = 2$ levels, and all leaf taxa are $x = 2$ branches away from the root.

Where $\mathscr{R}$ is the set of rows in the matrix of pairwise distances:

1. While $|\mathscr{R}| > 2$:
   (a) Choose a row $A \in \mathscr{R}$.
   (b) Choose a row $B \in \{\mathscr{R} \mid B \neq A\}$.
   (c) Calculate the set of transformed distances $T_A = \{T_{AR} \forall R \in \{\mathscr{R} | R \neq A\}\}$.
   (d) Calculate the set of transformed distances $T_B = \{T_{BR} \forall R \in \{\mathscr{R} | R \neq B\}\}$.
   (e) If $T_{AB} \in \{\min(T_A)\}$ and $T_{BA} \in \{\min(T_B)\}$:
      i Create a new node X, and join it to the nodes represented by A and B.
      ii Remove A and B from $\mathscr{R}$.
      iii Insert a row that corresponds to node X into $\mathscr{R}$.
2. Join the nodes represented by the remaining two rows.

There is a situation in which RNJ could mistakenly join two nodes, unless an additional check is performed. For example, nodes C and D in Fig. 1 could be mistakenly joined, since T_{CD} is minimal compared to the transformed distances in the matrix rows and columns that correspond to nodes C and D. However, there is a simple way to always recognize and avoid such errant joins when distances are additive. (For nonadditive distances, conflicting data lend partial support to such joins, so they are not necessarily errant.) Consider that if the path $A \leftrightarrow B$ includes an internal branch, there exists a node R such that the internal branch is a component of the path $A \leftrightarrow R$. In such a case, joining A and B would remove the internal branch, which would change T_{AR}. In fact, the way that branch lengths are calculated during joins changes the distance from A to all other nodes (except B) if there should be an internal branch between A and B. Therefore, when distances are additive, errant joins can be avoided by making sure that T_{AR} does not change for an arbitrary choice of R other than B.

RNJ's algorithmic complexity is $O(n^3)$, but typical performance is much better. The runtime of RNJ tree construction is primarily determined by the number of transformed distances that must be calculated. There is some probability that step (1a) of the RNJ algorithm will choose a taxon that has an immediate neighbor. That probability ranges from $4/n$ to 1; it is minimal for pectinate (maximally deep) trees such as that in Fig. 2 and maximal for perfect (fully balanced) trees such as that in Fig. 3. For the extreme case of pectinate trees, RNJ typically affords only a constant (though substantial) speedup over NJ, but for trees that are even somewhat balanced, RNJ performs approximately proportional to $n^2 \lg n$.

Experiments

We performed three experiments, which looked at (1) algorithmic correctness, (2) speed, and (3) quality of results. The correctness and speed experiments used additive distance matrices as input, in order to make validation possible and performance comparisons fair.

The experiments used four distinct tree shapes, and branch lengths were based on random sampling from the gamma distribution. The gamma distribution was chosen because it provides a simple mechanism for generating trees with varying degrees of branch length variation, where all branches have nonnegative lengths. In all cases, we chose parameters such that $\lambda = 1/\alpha$, where λ is the scale parameter and α is the shape parameter, so that the expected value was always 1. We chose values of $\alpha \geq 2$, so that all distributions had the same basic shape.

Specifics of how trees of the four shapes were generated follow:

Pectinate. A pectinate tree with x leaf taxa has $x - 1$ levels. For a given x, there is only one such tree shape, assuming unordered node adjacencies (Fig. 2). Each branch is assigned a length by summing the results of iteratively multiplying some constant branch length scalar C with a number that is drawn from the gamma distribution: $\sum_{i=L_{\min}}^{L_{\max}} C \times D_i$, where $D_i \sim \Gamma(\alpha, \lambda = 1/\alpha)$. L_{min} and L_{max} are the minimum and maximum levels that the branch spans. The root node is implicit, so the branch that contains the implicit root has a length that is the sum of that branch's implicit component branches.

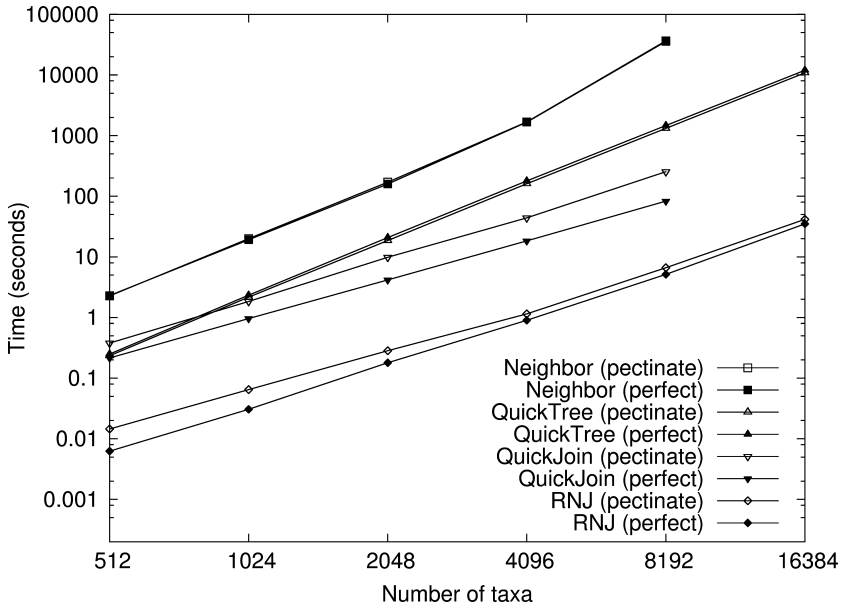

Fig. 4. Speed comparison of tree construction for two tree shapes (pectinate and perfect) and varying numbers of taxa. RNJ vastly outperforms the NJ implementations of QuickJoin, QuickTree, and PHYLIP neighbor for all tree shapes and sizes.

Treezilla. Treezilla trees are based on a single 500-taxon tree that is based on data from Chase et al. (1993). These trees differ only in their branch lengths. Each branch is assigned a length by multiplying some constant factor with a number that is drawn from the gamma distribution: $C \times D$, where $D \sim \Gamma(\alpha, \lambda = 1/\alpha)$.

Random. A random tree is generated via star decomposition. Each branch is assigned a length using the same procedure as for treezilla trees.

Perfect. A perfect tree always has 2^x leaf taxa, for some positive integer x. There are x levels of the tree, and every leaf taxon is x branches away from the root (Fig. 3). Each branch is assigned a length using the same procedure as for treezilla trees. The root node is implicit, so the branch that contains the implicit root is on average twice as long as the other branches. For purposes of tree generation, that branch's length is assigned as though it is two separate branches.

Correctness

We generated a total of 10 random trees for each of the following numbers of taxa: 3 to 50, and $100x$, where $x = [1..100]$. Branch lengths were gamma-distributed: $\sim \Gamma(\alpha = 2, \lambda = 1/2)$. For each of these trees we generated the corresponding additive distance matrix, then used RNJ to reconstruct a tree. In every case, RNJ succeeded in recovering the original tree.

Speed

We generated trees of the two extreme shapes: pectinate and perfect. Branch lengths were gamma-distributed: $\sim \Gamma(\alpha = 2, \lambda = 1/2)$. The trees had 2^x taxa, where $x = [9..14]$. From these trees, we generated additive distance matrices, which were randomly shuffled in order to avoid inputs that favored a particular search order. We then compared the runtime of RNJ to the runtimes of PHYLIP neighbor (Felsenstein 2004), QuickTree (Howe et al. 2002), and QuickJoin (Mailund and Pedersen 2004). The experiments were run on an Intel Pentium-4 3 GHz Linux system, and all four programs were compiled with the same optimization flags. Fig. 4 summarizes the results.

Quality

We used Rose (Stoye et al. 1998) to simulate true alignments and true trees for 512 taxa under the F84 model of molecular evolution (Kishino and Hasegawa 1989; Felsenstein and Churchill 1996) for four tree shapes (pectinate, treezilla, random, and perfect), ranges of sequence length (250 to 2500, in increments of 250), divergence time (0.25 to 2.5, in increments of 0.25), and level of evolutionary rate heterogeneity ($\alpha = 2^x$ for x from 1 to 10). Rose was run with a mean mutation rate of 0.01342302. Mutation rate is meaningful only in the context of time; we chose the mutation rate and time intervals such that the full range of useful divergence was simulated. We used insertion/deletion thresholds of 5.0×10^{-6}, and insertion/deletion function vectors of [.2,.3,.4,.4,.3,.2,.1]. We chose these insertion/deletion settings in order to add a level of biological realism to the problem of calculating pairwise sequence distances.

We used PHYLIP's dnadist program to estimate pairwise sequence distances according to the F84

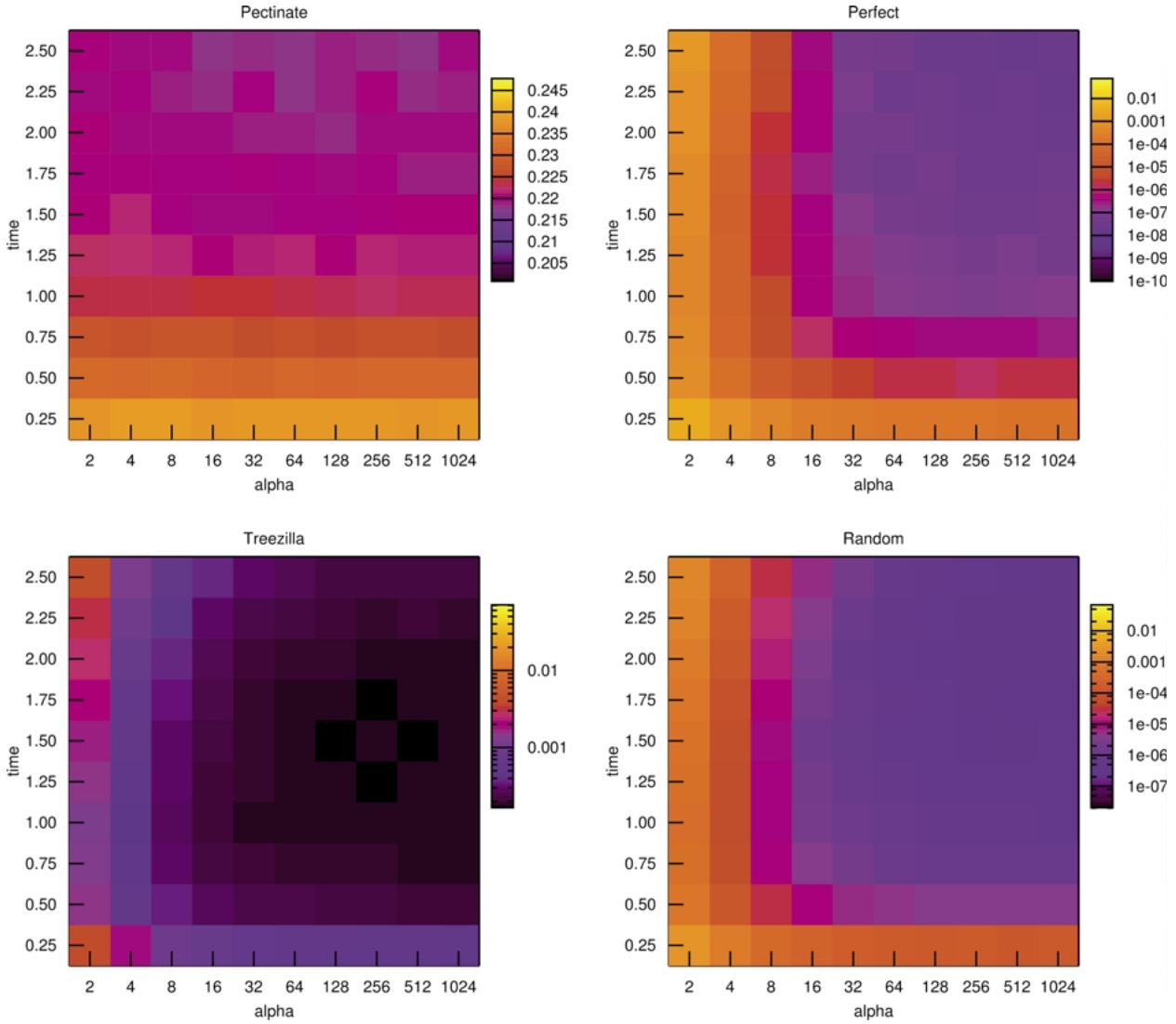

Fig. 5. RNJ quality results (mean MSE of Robinson-Foulds distances) for all four tree shapes (pectinate, treezilla, random, and perfect), where sequence length is 1500. The results are worst for pectinate trees and best for perfect trees.

model. We used QuickTree to generate NJ trees and used the Robinson-Foulds (1981) distance measure, specifically as described by Moret et al. (2004), to calculate the distance of each resulting tree from the corresponding true tree. The Robinson-Foulds distance measure represents the proportion of branches in two trees that induce bipartitions unique to one tree or the other.

Experiments were replicated on two levels.

1. One hundred RNJ and 100 NJ trees were created from each distance matrix, and the mean squared error (MSE) was calculated for the resulting Robinson-Foulds distances. The distance matrix was randomly shuffled before each NJ replicate, so that ties would be broken approximately randomly.
2. At a higher level, each experimental configuration was replicated 100 times and the mean and variance of the value mentioned in (1) were calculated. These two levels of replication were necessary in order to avoid spurious results due to stochasticity of RNJ, NJ, or the simulations. A total of 80 million trees were generated and analyzed in these tests. All of the quality experiments were run on a 220-processor Beowulf supercomputer.

Due to the large volume of the raw results, we were only able to fit representative samples here, along with descriptions of the general trends. In general, the quality of the RNJ and NJ results was very similar. Both algorithms performed best on perfect trees, nearly as well on random trees, somewhat worse on treezilla trees, and very poorly on pectinate trees.

Fig. 5 shows one slice of the results for RNJ. We do not present the variance of the mean squared error because for all experiments, a lower mean equated to a lower variance. The general patterns are very similar for RNJ and NJ; the only differences are variations in magnitude. As sequence length increased, results universally improved. Results were best for

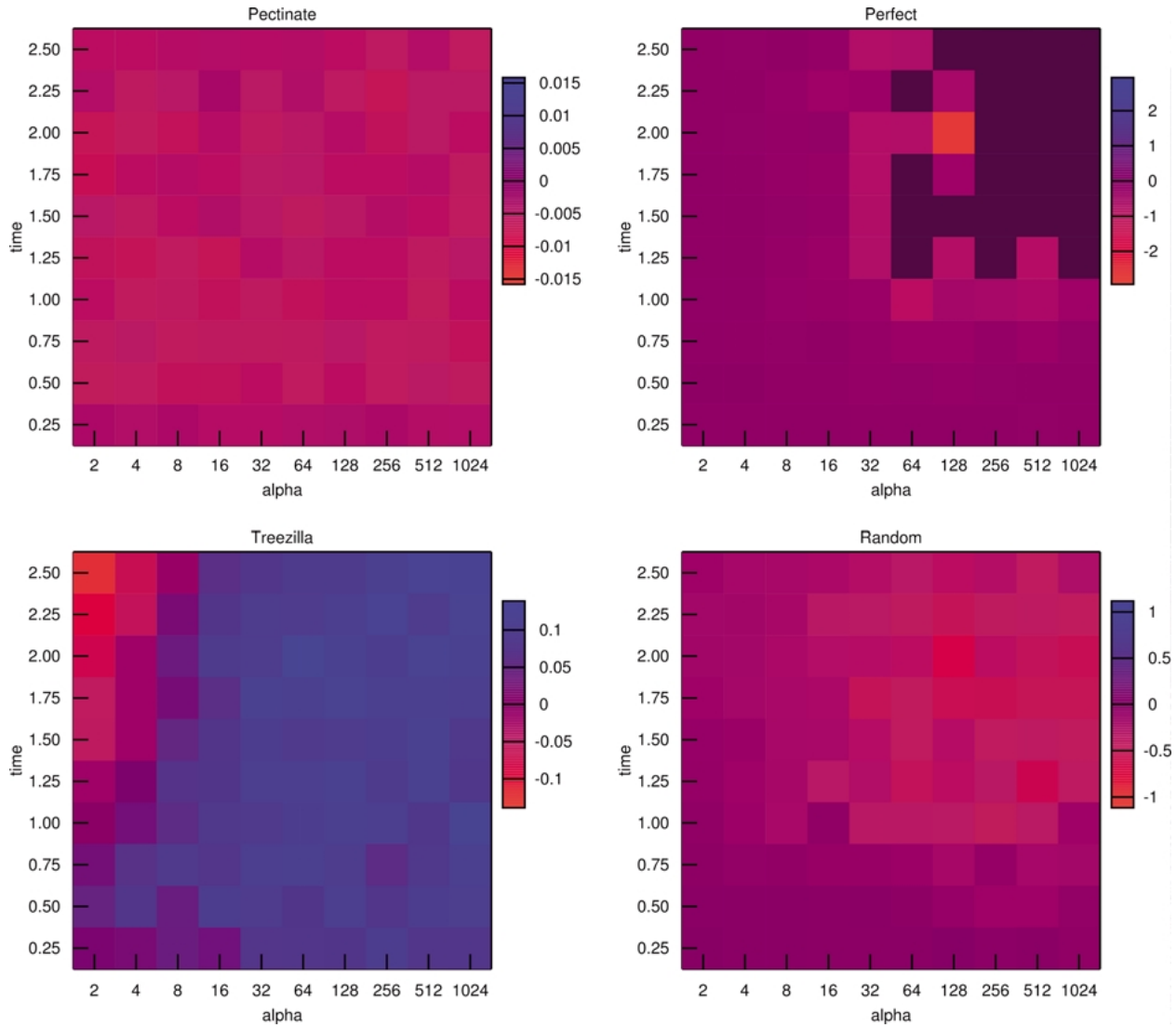

Fig. 6. Quality results of RNJ versus NJ, for all four tree shapes (pectinate, treezilla, random, and perfect), where sequence length is 1500. The plots depict relative quality using the formula $\log_{10} MMSE_{NJ} - \log_{10} MMSE_{RNJ}$, where $MMSE$ is the mean MSE of the Robinson-Foulds distances to the true trees. The formula calculates orders-of-magnitude differences between RNJ and NJ. Positive values mean that RNJ performed better than NJ. For the dark cells in the perfect trees plot, NJ always recovered the true tree topology.

$1.00 \leq time \leq 1.75$, depending on the tree shape. This was expected, because we chose the range of time such that at the extremes, PHYLIP dnadist was barely able to compute pairwise distances. As α increased (molecular clock rate heterogeneity decreased), results generally improved. For pectinate trees however, results were uniformly bad, regardless of α; RNJ and NJ generated poor trees because the variations in the long branches overwhelmed the short branches along the ''spines'' of the trees. This is an inherent aspect of pectinate trees, which calls into question the general usefulness of RNJ and NJ on pectinate trees. Nonetheless, we included pectinate trees because they represent one of the two extreme cases of tree topology, and because some previous NJ quality experiments included them (Saitou and Imanishi 1989).

Fig. 6 shows the relative performance of RNJ and NJ for one slice of the quality experiments. With few exceptions, NJ generated slightly better trees than RNJ did for all configurations of the experiments that were based on perfect, random, and pectinate trees. For the treezilla-based experiments, however, NJ only did slightly better for the shortest sequences, and the RNJ trees quickly surpassed those of NJ as the sequence length increased. It is important to note that even in this case, the algorithms produced trees of very similar quality.

Discussion

NJ favors joins for which there is maximal agreement about whether the nodes under consideration are

neighbors, as evidenced by transformed distances. RNJ treats all plausible joins as equally good. As such, RNJ trees tend to vary more than NJ trees. NJ is maximally greedy at each join, whereas RNJ is less greedy. NJ's greediness can cause systematic bias, which leaves open the risk of uniformly poor results for certain classes of input. Although RNJ is also potentially prone to bias, that bias is of a less troubling nature; all join operations for which the distance matrix contains support are given approximately equal opportunity, whereas NJ may completely exclude potential joins for which support is low. Were it possible to accurately quantify evidence for potential joins, an unbiased algorithm would randomly choose from the possibilities proportional to their levels of support.

The quality experiments were constructed such that the algorithm which performs better tends to have lower variance, because outliers have a large effect on the summary statistics. RNJ can construct a superset of the trees that NJ can construct, and this higher variance could be an advantage in some cases. Consider that the simulations used the same model of molecular evolution for both simulation and pairwise distance estimation. There was no mismatch between the true model and the inference model, so lower variance generally meant better overall results, but if there were a model mismatch, as is the case for biological data, RNJ's higher variance would improve the chances of capturing the true tree in its distribution of possible resulting trees. Thus RNJ is more robust than NJ in such a case.

We demonstrated that RNJ is substantially faster than NJ, which makes RNJ compelling for certain uses. For example, heuristic searches for optimal trees often start with NJ trees and try to improve from there. For large datasets, the substantial time that RNJ saves compared to NJ can instead be used for the heuristic search, which should allow more starting points to be considered in the same amount of time. Another application is that of guide tree creation for progressive multiple sequence alignment (MSA), as implemented by programs such as Clustal W (Thompson et al. 1994). NJ is the most algorithmically complex step of progressive MSA, so for large numbers of sequences, using RNJ instead of NJ can have a substantial impact on total program runtime.

As for overall quality of results, neither algorithm is clearly superior. NJ clearly produces better trees on average for perfect trees, but RNJ produces better trees on average for treezilla-based trees. That RNJ does better than NJ for the experiments that were based on biological data leaves us to wonder if this is peculiar to the treezilla data or if RNJ will generate better trees than NJ for a wide range of biologically based data.

Availability

The Clearcut implementation of RNJ is available upon request. Clearcut also includes an NJ implementation that is faster than any other nonheuristic implementations that the authors are aware of.

Acknowledgments. The authors thank Paul Joyce and Holly Wichman for their suggestions regarding experimental design, Bryan Carstens, Jack Sullivan, the students in the spring 2005 systematic biology course at the University of Idaho, and two anonymous reviewers for their review comments, and Bernard Moret for discussions regarding distance metrics. Evans and Foster were partially funded by NIH NCRR 1P20 RR16448. Sheneman was partially funded by NIH P20 RR16454 from the INBRE Program of the National Center for Research Resources. Some of the experiments were run on the IBEST Beowulf cluster, which is funded in part by NSF EPS 00809035, NIH NCRR 1P20 RR16448, and NIH NCRR 1P20 RR16454.

References

Bruno WJ, Socci ND, Halpern AL (2000) Weighted neighbor joining: a likelihood-based approach to distance-based phylogeny reconstruction. Mol Biol Evol 17:189–197

Chase MW, Soltis DE, Olmstead RG, Morgan D, Les DH, Mishler BD, Duvall MR, Price RA, Hills HG, Qiu YL, Kron KA, Rettig JH, Conti E, Palmer JD, Manhart JR, Sytsma KJ, Michaels HJ, Kress WJ, Karol KG, Clark WD, Hédren MH, Gaut BS, Jansen RK, Kim KJ, Wimpee CF, Smith JF, Furnier GR, Strauss SH, Xiang QY, Plunkett GM, Soltis PS, Swensen SM, Williams SE, Gadek PA, Quinn CJ, Equiarte LE, Dolenberg E, Learn GH Jr, Graham SW, Barrett SCH, Dayandan S, Albert VA (1993) Phylogenetics of seed plants: An analysis of nucleotide sequences from the plastid gene *rbc*L. Ann Mo Bot 80:528–580

Felsenstein J (2004) PHYLIP (Phylogeny Inference Package) version 3.6. Distributed by the author Department of Genome Sciences; University of Washington; Seattle

Felsenstein J, Churchill GA (1996) A hidden Markov Model approach to variation among sites in rate of evolution. Mol Biol Evol 13:93–104

Gascuel O (1997) BIONJ: an improved version of the NJ algorithm based on a simple model of sequence data. Mol Biol Evol 14:685–695

Howe K, Bateman A, Durbin R (2002) QuickTree: Building huge neighbour-joining trees of protein sequences. Bioinformatics 18:1546–1547

Kimura M (1980) A simple method for estimating evolutionary rates of base substitutions through comparative studies of nucleotide sequences. J Mol Evol 16:111–120

Kishino H, Hasegawa M (1989) Evaluation of the maximum likelihood estimate of the evolutionary tree topologies from DNA sequence data, and the branching order in Hominoidea. J Mol Evol 19:170–179

Kuhner MK, Felsenstein J (1994) A simulation comparison of phylogeny algorithms under equal and unequal evolutionary rates. Mol Biol Evol 11:459–468

Mailund T, Pedersen CNS (2004) QuickJoin—Fast neighbour-joining tree reconstruction. Bioinformatics 20:3261–3262

Moret BME, Nakhleh L, Warnow T, Linder CR, Tholse A, Padolina A, Sun J, Timme R (2004) Phylogenetic networks: modeling, reconstructibility, and accuracy. IEEE/ACM Trans Comput Biol Bioinform 1:13–23

Robinson DF, Foulds LR (1981) Comparison of phylogenetic trees. Math Bioscie 53:131–147

Saitou N, Imanishi T (1989) Relative efficiencies of the Fitch-Margoliash, maximum-parsimony, maximum-likelihood, minimum-evolution, and neighbor-joining methods of phylogenetic tree construction in obtaining the correct tree. Mol Biol Evol 6:514–525

Saitou N, Nei M (1987) The neighbor-joining method: a new method for reconstructing phylogenetic trees. Mol Biol Evol 4:406–425

Stoye J, Evers D, Meyer F (1998) Rose: generating sequence families. Bioinformatics 14:157–163

Studier JA, Keppler KJ (1988) A note on the neighbor-joining algorithm of Saitou and Nei. Mol Biol Evol 5:729–731

Thompson JD, Higgins DG, Gibson TJ (1994) CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. Nucleic Acids Res 22:4673–4680

Waterman MS, Smith TF, Singh M, Beyer WA (1977) Additive evolutionary trees. J Theor Biol 64:199–213