Theoretical Computer Science 410 (2009) 1993–2000

Contents lists available at ScienceDirect

Theoretical Computer Science

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

Fast neighbor joining☆

Isaac Elias *, Jens Lagergren

Department of Numerical Analysis and Computer Science, Royal Institute of Technology, Stockholm, Sweden

ARTICLE INFO

Article history:
Received 8 May 2006
Received in revised form 8 June 2007
Accepted 17 December 2008
Communicated by A. Apostolico

Keywords:
Neighbor joining
Phylogeny
Nearly additive
Fast neighbor joining
Pairwise string distance

ABSTRACT

Reconstructing the evolutionary history of a set of species is a fundamental problem in biology and methods for solving this problem are gaged based on two characteristics: accuracy and efficiency. Neighbor Joining (NJ) is a so-called distance-based method that, thanks to its good accuracy and speed, has been embraced by the phylogeny community. It takes the distances between n taxa and produces in $\Theta(n^3)$ time a phylogenetic tree, i.e., a tree which aims to describe the evolutionary history of the taxa. In addition to performing well in practice, the NJ algorithm has optimal reconstruction radius.

The contribution of this paper is twofold: (1) we present an algorithm called Fast Neighbor Joining (FNJ) with optimal reconstruction radius and optimal run time complexity $O(n^2)$ and (2) we present a greatly simplified proof for the correctness of NJ. Initial experiments show that FNJ in practice has almost the same accuracy as NJ, indicating that the property of optimal reconstruction radius has great importance to their good performance. Moreover, we show how improved running time can be achieved for computing the so-called correction formulas.

1. Introduction

The evolutionary history of a set of species is a central concept in biology that is commonly described by a phylogenetic tree. Frequently it is the case that the phylogenetic tree is unknown and the only information available are the genetic sequences from the extant species, i.e., currently living species. It is therefore a fundamental problem to reconstruct the phylogenetic tree given some genetic sequences. Several reconstruction methods have been suggested, and it is natural to compare these based on how accurate they are in reconstructing the correct phylogeny. Unfortunately, of these methods the more accurate are much too slow to be used in studies that involve reconstructing large or many phylogenies. The focus of this paper is to build an algorithm that is accurate and has quadratic running time in the number of species.

As more genetic information is collected it becomes possible to answer more complex questions. An obvious question that involves reconstructing a large phylogeny is to relate all living species in the tree of life. Another very central question is to relate large sets of genes and from such phylogenies draw conclusion about their function and origin. However, reconstruction of large phylogenies is not the only case in which efficient reconstruction is necessary. There are other cases that involve many reconstructions, e.g., studies where phylogenies are built for each gene shared by a set of species. The common technique of bootstrapping also requires many reconstructions in order to obtain significance values for a single phylogeny.

Throughout the paper, phylogenetic trees are leaf-labeled binary trees with edge lengths. Thus each phylogenetic tree T naturally induces an additive leaf-to-leaf distance function D_T. The reconstruction methods for which most complexity results have been shown are the so-called distance methods. These algorithms take as input an estimated distance function

☆ A preliminary version of this work has appeared in the proceedings of *ICALP 2005* [I. Elias, J. Lagergren, Fast neighbor joining, in: Proc. of the 32nd International Colloquium on Automata, Languages and Programming, ICALP'05, in: Lecture Notes in Computer Science, vol. 3580, Springer-Verlag, 2005, pp. 1263–1274].

* Corresponding address: KTH, Albanova, SBC, S - 100 44 Stockholm, Sweden. Tel.: +46 8 55 37 85 70; fax: +46 8 55 37 82 14.
E-mail addresses: isaac@nada.kth.se (I. Elias), jensl@nada.kth.se (J. Lagergren).

0304-3975/$ – see front matter © 2009 Published by Elsevier B.V.
doi:10.1016/j.tcs.2008.12.040

D (normally computed from the genomic sequences) and construct a phylogeny whose additive distance function is close to D. The problem of finding the closest additive distance function under the infinity norm is known to be NP-hard [1].

The Neighbor Joining (NJ) algorithm is a distance method introduced by Saitou and Nei in [18]. As shown in [10], when NJ is given an *additive* distance function D_T, it reconstructs the unique tree T. However, as Atteson [2] proved NJ reconstructs the closest tree for even more cases. A distance function D is *nearly additive* if there is an additive distance function D_T such that

$$|D - D_T|_\infty < \mu(T)/2, \tag{1}$$

where $\mu(T)$ is the minimum edge length in T. All the additive distance functions for which Eq. (1) holds have the same topology, i.e., disregarding the edge lengths, T is the unique tree for which the equation holds. The NJ algorithm has *optimal reconstruction radius* in the sense that: (a) given a nearly additive distance function it reconstructs the unique tree T and (b) there can be more than one tree for which $|D - D_T| < \delta$ holds if $\delta \geq \mu(T)/2$. In practice most distances are far from being nearly additive. Thus, although important, optimal reconstruction radius is not sufficient for an algorithm to be useful in practice.

The estimated distances that are given as input to distance methods are normally deduced from genomic sequences and a probabilistic model. There are various Markov models of sequence evolution which describe how sites evolve independently and identically from the root down toward the leafs. Many of these models have an associated closed correction formula for inverting the model and giving an estimated evolutionary distance for a pair of sequences. These formulas are consistent in the sense that the estimated distance approaches the underlying additive distance as the sequence length approaches infinity. As a result, the NJ algorithm is a consistent method for recovering the correct phylogeny, i.e., NJ reconstructs the correct phylogeny given some infinitely long sequences.

An interesting line of research is to design fast-converging algorithms, i.e., algorithms that reconstruct the correct phylogeny from sequences whose length is polynomial in the number of sequences [9,11,14,5]. However, except from the Disc-Covering Method (DCM) [11,14] these algorithms have had little or no practical impact. The only variation of DCM that is fast-converging and of practical interest uses NJ to construct small sub-phylogenies that are later patched together into one larger phylogeny, i.e., NJ is used as a subroutine.

Although the NJ algorithm is not fast-converging, it has been shown to perform very well in experimental studies [16]. Moreover, with $O(n^3)$ as the worst case running time it has become the reconstruction algorithm that is most frequently used in practice. Heuristic implementations of NJ have been given which, without leading to better worst case analysis of the time complexity, in practice show improved running time [3,20].

There are two major contributions in this paper [8]: (1) we present an algorithm called Fast Neighbor Joining (FNJ) with optimal reconstruction radius and optimal run time complexity $O(n^2)$ and (2) we present a greatly simplified proof for the correctness of the NJ algorithm. Initial experiments show that the FNJ algorithm in practice has almost the same accuracy as the NJ algorithm; this indicates that it is the optimal reconstruction radius and other similarities with NJ that give FNJ its good performance. We also describe how a better running time for computing the correction formulas can be achieved, in theory, through matrix multiplication and, in practice, through table lookups.

The FNJ algorithm is useful in its own right. But it is also important to note that FNJ together with the proof of optimal reconstruction radius presents a good foundation for building reconstruction algorithms that are both practically useful and fast-converging. For example the running time of DCM can be improved by a factor $O(n)$ by simply replacing NJ with FNJ. It will be interesting to see how the running time of extensions of NJ, such as Weighbor and BioNJ, can be improved using our ideas.

Since the publication of the preliminary version of this paper there has been two papers improving on the results of this paper. In [15], a relation between NJ and FNJ, on the one hand, and quartet methods, on the other hand, is given. This relation is used to prove that both NJ and FNJ have edge reconstruction radius 1/4. That is, if the maximal error in the estimated distances is $< \varepsilon/4$ then all edges of length $\geq \varepsilon$ are reconstructed correctly by both NJ and FNJ. Moreover, in [7], an advanced bit-fiddling algorithm for computing correction formulas is provided which in practice is a factor 400 faster than other available software.

The paper is organized as follows. The next section contains some basic definitions and a description of the NJ algorithm. In Section 3, the FNJ algorithm is introduced. Subsequently we give the proof of the FNJ algorithm and also a more economical and intuitively appealing proof of Atteson's theorem. Finally, in Section 7, we approach the practical problem of computing the correction formulas and also show that the FNJ algorithm in practice performs almost exactly as good as the NJ algorithm. Except for Lemma 2 below, which in [2] (Lemma 12) is proved by straightforward algebraic verification, the present paper is self-contained.

2. Definitions and the neighbor joining algorithm

A $n \times n$ *distance function* D, for a set of taxa $\mathcal{N}(D)$, is a function $\mathcal{N}(D)^2 \to R^+$, where $|\mathcal{N}(D)| = n$, which is symmetric and satisfies $D(x, x) = 0$ for every $x \in \mathcal{N}(D)$. For two distance functions D_1 and D_2 such that $\mathcal{N}(D_1) = \mathcal{N}(D_2) = \mathcal{N}$, their distance is defined as $\max_{x,y\in\mathcal{N}} |D_1(x, y) - D_2(x, y)|$ and denoted $|D_1 - D_2|_\infty$. By a *phylogenetic tree* we mean a tree T given together with an edge length function $l_T : E(T) \to R^+$. For a phylogenetic tree T, $\mu(T)$ denotes the minimum edge length of T, i.e., $\min_{e\in E(T)} l(e)$. The unique path in a tree T between two of its vertices u and v is denoted $P_T(u, v)$. Every phylogenetic tree T *induces* a distance function for the leafs in the tree, i.e., $D_T : L(T)^2 \to R^+$ where $D_T(a, b) \triangleq \sum_{e\in P_T(a,b)} l(e)$.

A distance function D is *additive* if there is a phylogenetic tree T such that $D = D_T$; the tree is said to *realize* D, it is unique, and it is denoted $T(D)$. A distance function D is *nearly additive* if there is a phylogenetic tree T such that $|D - D_T|_\infty < \mu(T)/2$; again, the tree is said to *realize* D, it is unique, and it is denoted $T(D)$ [2]. The *parent* of a leaf a in a tree T is the unique neighbor of a in T. A pair of leaves of a tree T are *siblings* if they have the same parent in T (note only leaf-siblings).

The NJ algorithm builds a tree by iteratively combining pairs of taxa. It takes as input a distance function D for n taxa and attempts to identify two siblings by selecting the pair of taxa (a, b) that minimizes the *NJ function*, defined by

$$S_D(x, y) \triangleq \big(|\mathcal{N}(D)| - 2\big) \cdot D(x, y) - \sum_{z \in \mathcal{N}(D)} \big(D(z, x) + D(z, y)\big). \tag{2}$$

Thereafter the pair (a, b) is *reduced* to a new node c, representing the parent, which gives a new distance function D' with $\mathcal{N}(D') = (\mathcal{N}(D) \setminus \{a, b\}) \cup \{c\}$ defined by

$$D'(x, y) \triangleq \begin{cases} D(x, y), & \text{if } c \notin \{x, y\} \\ \frac{D(z,a)+D(z,b)}{2}, & \text{otherwise } z \in \{x, y\} \setminus \{c\}. \end{cases} \tag{3}$$

Finally the algorithm is applied iteratively on the new distance function D'. A formal description of the NJ algorithm is given

Algorithm NJ($\mathbf{D_1}$)

(1) For each $i \leftarrow 1$ to $n - 3$ do
  (a) $(a_i, b_i) \leftarrow \operatorname{argmin}_{x \neq y \in \mathcal{N}(D_i)} S_{D_i}(x, y)$
  (b) Reduce a_i and b_i to a new node c_i and let D_{i+1} be the new distance function given by the reduction in Eq. (3).
  (c) Connect a_i and b_i to c_i by adding edges (a_i, c_i) and (b_i, c_i).
(2) Connect the three nodes of $\mathcal{N}(D_{n-3})$ in a star and return the resulting tree.

Theorem 1 (*Atteson's Theorem*). *Given a nearly additive distance function D NJ outputs $T(D)$. Moreover, in each iteration i, D_i is nearly additive and $T(D_i) = T(D_{i-1}) \setminus \{a_{i-1}, b_{i-1}\}$.*

In Section 3, we will show the analogous theorem for the FNJ algorithm, by showing that for nearly additive distance functions it gives exactly the same output as NJ. In Section 5, we give a proof of Atteson's theorem above.

3. The fast neighbor joining algorithm

In Step 1a of the NJ algorithm and for $i \leq n/2$, the minimum is taken over $\Omega(n^2)$ pairs which implies a running time of $\Omega(n^3)$. In the FNJ algorithm an $O(n^2)$ running time is obtained by using two ideas. First, the minimum is taken over a set, called the *visible set*, of cardinality $O(n)$. Second, using the auxiliary function R, introduced below, the updated NJ function can be computed in constant time. It should be noted that the resulting trees of NJ and FNJ are only guaranteed to be the same if the input is nearly additive.

A pair (a, b) is *visible from* a w.r.t. a distance function D if

$$b = \operatorname*{argmin}_{x \in \mathcal{N}(D) \setminus \{a\}} S_D(a, x).$$

A pair (a, b) is *visible* w.r.t. D if it is visible from either a or b. Hence the number of visible pairs is $O(n)$. In the next section it is shown that for each nearly additive distance function D, any sibling pair in $T(D)$ is visible w.r.t. D.

To enable an overall $O(n^2)$ running time, the NJ function is computed using an auxiliary function R defined by $R_D(a) \triangleq \sum_{x \in \mathcal{N}(D)} D(a, x)$, i.e., R is the row sums. It is straightforward to verify that for a D' defined as in Eq. (3),

$$R_{D'}(x) = R_D(x) - \frac{D(x, a) + D(x, b)}{2}. \tag{4}$$

Hence, given R_D it is possible to compute the updated row sums $R_{D'}$ in time $O(n)$. Moreover, since $S_D(x, y) = (|\mathcal{N}(D)| - 2) \cdot D(x, y) - R_D(x) - R_D(y)$, the NJ function can be computed in constant time, for any given pair (x, y).

It should be clear, from the formal description below that the FNJ algorithm runs in time $O(n^2)$. Note that the input actually has size $\Omega(n^2)$.

Algorithm FNJ($\mathbf{D_1}$)

(1) The first visible set $\mathcal{V}_1$ is initialized to the set of pairs visible w.r.t. D_1.
(2) For each $a \in \mathcal{N}(D)$, $R_{D_1}(a)$ is initialized to $\sum_{x \in \mathcal{N}(D_1)} D_1(a, x)$.
(3) For each $i \leftarrow 1$ to $n - 3$ do
  (a) $(a_i, b_i) \leftarrow \operatorname{argmin}_{(x,y) \in \mathcal{V}_i} S_{D_i}(x, y)$
  (b) Reduce a_i and b_i to a new node c_i and let D_{i+1} be the new distance function given by the reduction in Eq. (3).
  (c) Connect a_i and b_i to c_i by adding edges (a_i, c_i) and (b_i, c_i).
  (d) Compute $R_{D_{i+1}}$.
  (e) $\mathcal{V}_{i+1} \leftarrow (\mathcal{V}_i \setminus \{(x, y) : x = a_i \text{ or } x = b_i\}) \cup \{(c_i, d)\}$ where (c_i, d) is the pair visible from c_i w.r.t. D_{i+1}.
(4) Connect the three nodes of $\mathcal{N}(D_{n-3})$ in a star and return the resulting tree.

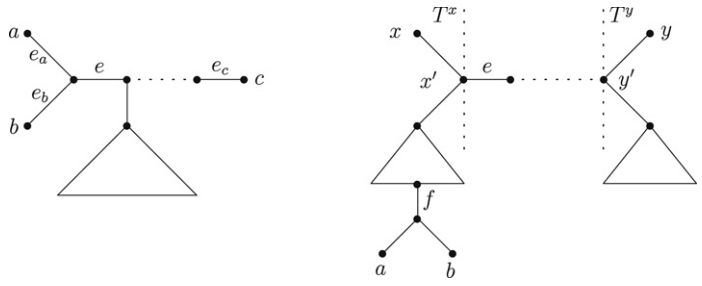

Fig. 1. To the left the figure for the Visibility lemma. To the right the figure for Lemma 5.

4. Correctness of FNJ

According to Theorem 1, given a nearly additive distance function D, NJ outputs $T(D)$, i.e., it outputs the unique tree that is close to D. Here we prove that FNJ has the same property. Since NJ constructs the correct tree, we know that in each iteration the minimum pair over the NJ function is a sibling pair in $T(D_i)$. Hence, to prove the correctness of FNJ, it suffices to show that in each iteration the minimum pair is in the visible set, $\mathcal{V}_i$. The proof is in two steps; first the Visibility lemma is presented. According to this, if a has a sibling b in $T(D)$, then (a, b) is in the visible set. Second, in Theorem 4, the Visibility lemma together with the correctness of NJ is used to prove the correctness of FNJ.

Before we proceed to prove the Visibility lemma, we state an observation and a lemma which in Atteson [2] are proved through straightforward algebraic verification. For any tree T, edge e of T, and leaf a of T, let $\mathcal{L}_T(a, e)$ denote the set of leaves of T belonging to the same connected component of $T \setminus \{e\}$ as a.

Observation 1 (*Atteson*). *If D_T is an additive distance function, then,*

$$S_{D_T}(a, b) = \sum_{e \in E(T)} w_e(a, b)\, l(e), \text{ where}$$

$$w_e(a, b) = \begin{cases} -2 & \text{if } e \in E(P_T(a, b)) \\ -2|L(T) \setminus \mathcal{L}_T(a, e)| & \text{otherwise.} \end{cases}$$

Lemma 2 (*Atteson, Lemma 12*). *Let D_T and D be two n-domain distance functions such that D_T is additive and D is nearly additive w.r.t. D_T. For any $a, b, x, y \in \mathcal{N}(D)$, the value of $S_D(a, b) - S_{D_T}(a, b) + S_{D_T}(x, y) - S_D(x, y)$ is*

$$> \begin{cases} -3(n-4)\mu(T) & \text{if } \{a, b\} \cap \{x, y\} = \emptyset \\ -2(n-3)\mu(T) & \text{if } |\{a, b\} \cap \{x, y\}| = 1. \end{cases}$$

Lemma 3 (*The Visibility Lemma*). *Let D_T and D be two n-domain distance functions such that D_T is additive and D is nearly additive w.r.t. D_T. If a has a sibling b in T, then (a, b) is visible from a w.r.t. D, i.e.,*

$$b = \operatorname*{argmin}_{x \in \mathcal{N}(D) \setminus \{a\}} S_D(a, x).$$

Proof. As in Fig. 1, let $c \in \mathcal{N}(D) \setminus \{a, b\}$ and let e_a, e_b, and e_c be the edges of T incident with a, b, and c, respectively. Moreover, let e be the edge incident with the parent of a and b which is not incident with either a or b. Consider D_T, by definition of the weights in Observation 1 the following is true

(i) $w_f(a, b) = -2 = w_f(a, c)$ for any $f \in \{e_a, e_b, e_c\}$,
(ii) $w_f(a, b) \leq -3 < w_f(a, c)$ for any $f \in E(P_T(a, c)) \setminus \{e_a, e_c\}$,
(iii) $w_f(a, b) = w_f(a, c)$ for any $f \in E(T) \setminus E(P_T(a, c))$.

Moreover, since $w_e(a, b) = -2(n-2)$ and $w_e(a, c) = -2$, it follows that $S_{D_T}(a, c) - S_{D_T}(a, b) \geq 2(n-3)\mu(T)$. Finally, by Lemma 2,

$$S_D(a, c) - S_D(a, b) = \underbrace{S_D(a, c) - S_{D_T}(a, c) + S_{D_T}(a, b) - S_D(a, b)}_{> -2(n-3)\mu(T)} + \underbrace{S_{D_T}(a, c) - S_{D_T}(a, b)}_{\geq 2(n-3)\mu(T)} > 0. \quad \square$$

We are now ready to prove that given a nearly additive distance function D, FNJ in each iteration selects the same sibling pair as NJ, i.e., FNJ outputs $T(D)$. By Atteson's theorem NJ outputs $T(D)$ in each iteration by reducing a pair of siblings such that $T(D_i) = T(D_{i-1}) \setminus \{a_{i-1}, b_{i-1}\}$. Since FNJ uses the same reduction as NJ it is sufficient to show that all sibling pairs are in the visible set. In the next section, we give a short and intuitively appealing proof of Atteson's theorem, which together with the Visibility lemma gives a direct proof of the theorem below.

Theorem 4. *Given a nearly additive distance function D, FNJ outputs $T(D)$.*

Proof. We prove by induction that, for each $i = 1, \ldots, n-3$, $\mathcal{V}_i$ contains all sibling pairs of $T(D_i)$ (here $D_1 = D$). By the Visibility lemma it is clear that the statement is true for $i = 1$. Assume that the statement holds for each $i = 1, \ldots, j$.

By the correctness of NJ, if (a_j, b_j) is the minimum over the NJ function, then (a_j, b_j) is a sibling pair $T(D_j)$. Therefore, by the induction assumption, (a_j, b_j) is in $\mathcal{V}_j$. Consequently, since the minimum over the NJ function is a sibling pair, FNJ and NJ select the same sibling pair in iteration j.

After reducing (a_j, b_j) to c_j, by the Visibility lemma, if c_j has a sibling d in $T(D_{j+1})$, then in Step 3e (c_j, d) is added to $\mathcal{V}_{j+1}$. Moreover, by the assumption, all other sibling pairs of $T(D_{j+1})$ are in $\mathcal{V}_j$ and therefore also in $\mathcal{V}_{j+1}$. Hence, by induction and the correctness of NJ, FNJ outputs $T(D)$. □

5. Atteson's theorem — correctness of NJ

The proof of Atteson's theorem is in two steps. The first step consists of the key technical lemma below, of which we give a much more concise and direct proof. The central idea in this proof, is to show that for any additive distance function the difference is large between the value of NJ function applied to a sibling pair, and applied to a pair of leaves which are not siblings. In fact, the difference is so large that even when the distance function is nearly additive the NJ function is minimized by a sibling pair. The final step in proving Atteson's theorem consists of showing that the distance function, after a reduction, remains nearly additive.

Lemma 5. *If D is a nearly additive distance function, $a, b \in \mathcal{N}(D)$, and $S_D(a, b) = \min_{x \neq y \in \mathcal{N}(D)} S_D(x, y)$, then (a, b) is a sibling pair in $T = T(D)$.*

Proof. According to the Visibility lemma, if a has a sibling b then $S_D(a, b) < S_D(a, x)$ for any $x \neq b$. Hence, the lemma follows if for any two leaves, x and y, of which none has a sibling in T, there exists a sibling pair (a, b), such that $S_D(x, y) - S_D(a, b) > 0$. Let D_T be an additive distance function such that $|D - D_T| < \mu(T)/2$. Notice that

$$\begin{aligned} S_D(x, y) - S_D(a, b) &= S_D(x, y) - S_{D_T}(x, y) + S_{D_T}(a, b) - S_D(a, b) + S_{D_T}(x, y) - S_{D_T}(a, b) \\ &> -3(n-4)\mu(T) + S_{D_T}(x, y) - S_{D_T}(a, b), \end{aligned}$$

where the inequality follows by Lemma 2. We proceed by showing that $S_{D_T}(x, y) - S_{D_T}(a, b) > 3(n-4)\mu(T)$.

In T let x' and y' be the unique neighbors of x and y, respectively (see Fig. 1). Further, let T^x and T^y be the subtrees of $T \setminus P_T(x', y')$ containing x and y, respectively. W.l.o.g., assume that $|L(T^x)| \leq |L(T^y)|$, and hence that $|L(T^x)| \leq n/2$. Let e be the edge of $P_T(x', y')$ incident to x'. Since neither x nor y has a sibling, both T^x and T^y contain a sibling pair of T. Let a and b be siblings in T^x, and let f be the edge incident with their parent but not a and not b.

First note that $w_g(a, b) \leq w_g(x, y)$ for any $g \in E(T) \setminus \{e, f\}$. The only edges for which the latter inequality is non-trivial are those of $P_T(a, x')$; for those the inequality follows from the assumption that $|L(T^x)| \leq n/2$. Using the definition of weights, it is straightforward to verify that $w_e(a, b) = -2|L(T) \setminus L(T^x)| \geq -n$ while $w_e(x, y) = -2$, and that $w_f(a, b) = -2(n-2)$ while $w_f(x, y) = -4$. It follows that

$$\begin{aligned} S_{D_T}(x, y) - S_{D_T}(a, b) &\geq \big(-2 - 4 + n + 2(n-2)\big)\mu(T) \\ &= (3n - 10)\mu(T) \\ &> 3(n-4)\mu(T). \quad \square \end{aligned}$$

Proof of Theorem 1. The proof is by induction. First note that the theorem holds when $|\mathcal{N}(D)| = 3$. Assume that the theorem holds when $|\mathcal{N}(D)| = n - 1$. We now prove that it holds for $|\mathcal{N}(D)| = n$.

Since D is nearly additive, by the lemma above, NJ in the first iteration reduces a pair (a, b) that are siblings in $T = T(D)$ to a new node c, representing their parent. Denote the distance function after the reduction by D'. We need to prove that D' is nearly additive and that $T(D') = T \setminus \{a, b\}$.

Let S be the tree $T \setminus \{a, b\}$ with the edge length function defined as follows:

$$l_S(u, v) \triangleq l_T(u, v)$$

for all $u, v \in V(S) \setminus \{c\}$, and

$$l_S(c, c') \triangleq l_T(c, c') + \frac{l_T(c, a) + l_T(c, b)}{2}$$

for the unique neighbor c' of c in S. It should be clear that $\mu(T) \leq \mu(S)$.

We now show that $|D' - D_S| < \mu(S)/2$, i.e., that $T(D') = S = T \setminus \{a, b\}$. From this, the theorem follows immediately. For $u, v \in L(S) \setminus \{c\}$,

$$|D'(u, v) - D_S(u, v)| = |D(u, v) - D_T(u, v))| < \frac{\mu(T)}{2} \leq \frac{\mu(S)}{2}.$$

For all $u \in L(S)$,

$$\begin{aligned} \big|D'(u, c) - D_S(u, c)\big| &= \left|\frac{D(u, a) + D(u, b)}{2} - D_S(u, c') - l_S(c', c)\right| \\ &= \left|\frac{D(u, a) + D(u, b)}{2} - D_T(u, c') - l_T(c', c) - \frac{l_T(c, a) + l_T(c, b)}{2}\right| \end{aligned}$$

$$\leq \left|\frac{D(u,a) - D_T(u,a)}{2} + \frac{D(u,b) - D_T(u,b)}{2}\right|$$
$$\leq \left|\frac{D(u,a) - D_T(u,a)}{2}\right| + \left|\frac{D(u,b) - D_T(u,b)}{2}\right|$$
$$< \frac{\mu(T)}{4} + \frac{\mu(T)}{4} = \frac{\mu(T)}{2} \leq \frac{\mu(S)}{2}. \quad \square$$

6. Improved computations of correction formulas

As was mentioned in the introduction, the real input to a reconstruction problem is usually n sequences of length l. The assumption is that these sequences have evolved from an original ancestor sequence down the branches of the phylogeny, according to a model of sequence evolution. The distance method approach, to the reconstruction problem, is to first use the sequences to estimate the actual distances between every pair of leaves, and thereafter find a phylogeny that fits the estimated distances. That is, from the n sequences of length l, an $n \times n$ distance function is computed through a correction formula. This formula is dependent on the model assumed to have generated the sequences; the most common models are Jukes–Cantor (JC) [12] and Kimura 2-parameter (K2P) [13]. Most correction formulas are in a sense functions of the hamming distance, e.g., the JC correction formula is given by

$$JC(s_1, s_2) \triangleq -\frac{3}{4} \cdot \log\left(1 - \frac{4 \cdot H(s_1, s_2)}{3l}\right),$$

where H is the hamming distance. Clearly, the straightforward way of computing this function takes $O(l)$ time, and as a result the overall running time of computing all estimated distances is $O(ln^2)$. Since l typically is larger than n, the computation of the correction formula is the bottleneck in fast reconstruction algorithms.

Computing all n^2 pairwise hamming distances for n strings is a special case of matrix multiplication, and can therefore be done in $O(ln^{1.376})$ time [4]. The reduction for strings from the alphabet $\{A, C, G, T\}$, is by representing each string by a row in the matrix M, and code each symbol by the unary code, e.g., by letting $A = 1000$. Thereby, the elements in the matrix MM^T are $l - H(s_i, s_j)$. It should be noted that the general belief is that matrix multiplication can be done in $O(ln)$ time, which would imply that the correction formulas can be computed in optimal time. Unfortunately, all existing matrix multiplication algorithms are slow in practice.

Below we present an algorithm that improved the computations of the correction formula by more than a factor of 3, compared to the straightforward approach. The idea is to first represent each symbol by 2 bits, and then use a precomputed table with 2^{2k} entries to look up the distance for k symbols at a time. In our tests, $k = 7$ resulted in the best running time.

(1) Code the symbols of the sequences as follows: $A = 00$, $C = 01$, $G = 10$, $T = 11$.
(2) For each pair of compacted strings c_i and c_j
  (a) Compute the xor $X_{ij} = c_i \bigoplus c_j$.
  (b) Read $2k$ bits of X_{ij} at a time and use the table to look up the distance for the associated k symbols.

7. Experiments

In this paper it has been shown that both NJ and FNJ have optimal reconstruction radius. However, there are many distance matrices that are not nearly additive and for which both algorithms reconstruct the closest tree. And for yet more matrices the algorithms fail to reconstruct the tree, but they do not fail by much. Therefore, it is of major interest to know how well the two algorithms perform in practice.

Several studies have been made on the accuracy of different reconstruction algorithms, the most notable work being that by Nakhleh *et al.* [16]. In that paper, four different methods are examined: NJ, DCM-NJ + MP, Weighbor, and Greedy Parsimony. And it is noted that the NJ algorithm, because of its speed, is the method of choice when the input data are accurate, i.e., when the sequence length is large and the corrected distances are close to additive. In this section, we replicate some of the experiments and show that although the NJ algorithm perform slightly better than the FNJ algorithm, when the input data are accurate the performance is in fact close to the same.

The test data were produced in the same way as in [16]. First, the model trees were generated through a random birth–death process using the `r8s` [19] software package. These trees where then made non-ultrametric, i.e., root to leaf paths where made to vary in length, by multiplying the edge lengths with a random number in different intervals.[1] Subsequently, sequence data was generated according to the JC model using the `Seq-Gen` [17] program. The JC correction formula was then applied to get the distances, and for saturated data a fix factor of 1 was used.

[1] Following [16] we used ultrametric deviation 4 and generated sequences with diameter factors 0.05, 0.10, 0.25, and 0.5. E.g. diameter factor 0.25 yields the interval $[1/16, 1]$.

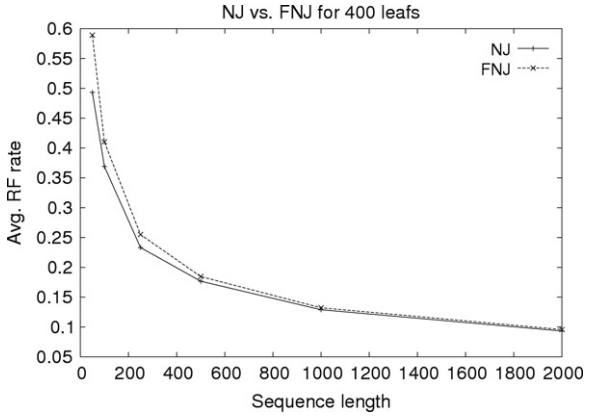

Fig. 2. Comparing the accuracy of NJ vs. FNJ for trees with 400 taxa and varying sequence length.

Table 1
Comparing the running time and accuracy of NJ, FNJ, and GME, for 10 trees of 4000 taxa.

Algo.	Time (min)	Avg. RF (%)
FNJ	4	10.5
NJ	52	10.1
GME	26	14.1

To measure the accuracy we used the normalized Robinson–Foulds (RF) distance between the model tree and the tree given by the method. To get statistically robust results we performed 20 runs on each test size, and computed the average RF rate and standard deviation. In Fig. 2 we plot the average RF rate as a function of the sequence length for trees with 400 taxa. Notice that both methods converge to the true tree as the sequence length increases, and that for accurate data the methods perform almost the same. For these experiments the standard deviation varied between 1%–4% except for sequences of length 50. Many more experiments have been performed and the same pattern emerges there too but due to space limitations these data have been omitted.

7.1. Comparison with GME

In Desper *et al.* [6], an $O(n^2)$ algorithm called GME is introduced that, although it does not have optimal reconstruction radius, in practice it has acceptable accuracy. However, as is clearly shown in Table 1, for 10 trees of 4000 taxa each, FNJ outperforms both GME and NJ. When accuracy is concerned the best algorithm is NJ, tightly followed by FNJ. In addition to GME, Desper *et al.* present a clever nearest neighbor interchange (NNI) algorithm, that in many cases improves the accuracy of reconstruction algorithms. It is therefore reasonable to believe that FNJ in conjunction with NNI would be a very fast and accurate combination.

Acknowledgments

We would like to thank Luay Nakhleh and Tandy Warnow for discussions on experimental studies of the NJ algorithm and for helping us replicate their experiments. We are also grateful to Johan Håstad for valuable comments and ideas.

References

[1] R. Agarwala, V. Bafna, M. Farach, M. Paterson, M. Thorup, On the approximability of numerical taxonomy (fitting distances by tree metrics), SICOMP 28 (3) (1999) 1073–1085.
[2] K. Atteson, The performance of neighbor-joining methods of phylogenetic reconstruction, Algorithmica 25 (1999).
[3] G.S. Brodal, R. Fagerberg, T. Mailund, C.N. Pedersen, D. Phillips, Speeding up neighbour-joining tree construction, Technical Report ALCOMFT-TR-03-102, 2003.
[4] D. Coppersmith, S. Winograd, Matrix multiplication via arithmetic progressions, in: STOC'87, 1987, pp. 1–6.
[5] M. Csűrös, Fast recovery of evolutionary trees with thousands of nodes, in: RECOMB-01, pp. 104–113, 2001.
[6] R. Desper, O. Gascuel, Fast and accurate phylogeny reconstruction algorithms based on the minimum-evolution principle, Journal of Computational Biology 19 (5) (2002) 687–705.
[7] I. Elias, J. Lagergren, Fast computation of distance estimators, BMC Bioinformatics 8 (2007) 89.
[8] I. Elias, J. Lagergren, Fast neighbor joining, in: Proc. of the 32nd International Colloquium on Automata, Languages and Programming, ICALP'05, in: Lecture Notes in Computer Science, vol. 3580, Springer-Verlag, July 2005, pp. 1263–1274.

[9] P.L. Erdös, M.A. Steel, L.A. Szekely, T.J. Warnow, A few logs suffice to build (almost) all trees (I), Random Structures & Algorithms 14 (1999) 153–184.
[10] O. Gascuel, Concerning the NJ algorithm and its unweighted version, in: UNJ, American Mathematical Society, 1997, pp. 149–170.
[11] D.H. Huson, S. Nettles, T. Warnow, Disk-covering, a fast-converging method for phylogenetic tree reconstruction, Journal of Computational Biology 6 (3/4) (1999) 369–386.
[12] T.H. Jukes, C.R. Cantor, Evolution of protein molecules, Mammalian Protein Metabolism (1969) 21–132.
[13] M. Kimura, A simple model for estimating evolutionary rates of base substitutions through comparative studies of nucleotide sequences, Journal of Molecular Evolution 16 (1980) 111–120.
[14] J. Lagergren, Combining polynomial running time and fast convergence for the disk-covering method, Journal of Computer and System Sciences 65 (2002).
[15] R. Mihaescu, D. Levy, L. Pachter, Why neighbor-joining works, Journal Algorithmica (2006), doi: 10.1007/s00453-007-9116-4. Published online http://www.springerlink.com/content/w3206717365g8m01/.
[16] L. Nakhleh, B.M.E. Moret, K. St John, J. Sun, U. Roshan, T. Warnow, The accuracy of fast phylogenetic methods for large datasets, in: PSB-02, 2002, pp. 211–222.
[17] A. Rambaut, N.C. Grassly, Seq-gen: An application for the Monte Carlo simulation of DNA sequence evolution along phylogenetic trees, Computer Applications in Biosciences 13 (1997) 235–238.
[18] N. Saitou, M. Nei, The neighbor-joining method: A new method for reconstructing phylogenetic trees, Molecular Biology and Evolution 4 (1987) 406–425.
[19] M. Sanderson, r8s software package. http://ginger.ucdavis.edu/r8s/.
[20] CN. Pedersen, T. Mailund, Quickjoin-fast neighbour-joining tree reconstruction, Bioinformatics (2004).