GMR Genetics and Molecular Research
Online Journal
ISSN:1676-5680

Methodology

FastJoin, an improved neighbor-joining algorithm

J. Wang, M.-Z. Guo and L.L. Xing

School of Computer Science and Technology, Harbin Institute of Technology, Harbin, Heilongjiang, P.R. China

Corresponding author: J. Wang
E-mail: wangjuanangle@hit.edu.cn

Genet. Mol. Res. 11 (3): 1909-1922 (2012)
Received January 10, 2012
Accepted March 12, 2012
Published July 19, 2012
DOI http://dx.doi.org/10.4238/2012.July.19.10

ABSTRACT. Reconstructing the evolutionary history of a set of species is an elementary problem in biology, and methods for solving this problem are evaluated based on two characteristics: accuracy and efficiency. Neighbor-joining reconstructs phylogenetic trees by iteratively picking a pair of nodes to merge as a new node until only one node remains; due to its good accuracy and speed, it has been embraced by the phylogeny research community. With the advent of large amounts of data, improved fast and precise methods for reconstructing evolutionary trees have become necessary. We improved the neighbor-joining algorithm by iteratively picking two pairs of nodes and merging as two new nodes, until only one node remains. We found that another pair of true neighbors could be chosen to merge as a new node besides the pair of true neighbors chosen by the criterion of the neighbor-joining method, in each iteration of the clustering procedure for the purely additive tree. These new neighbors will be selected by another iteration of the neighbor-joining method, so that they provide an improved neighbor-joining algorithm, by iteratively picking two pairs of nodes to merge as two new nodes until only one node remains, constructing the same phylogenetic tree as the neighbor-joining algorithm for the same input data. By combining the improved neighbor-joining algorithm with styles upper bound computation optimization of RapidNJ and external storage of ERapidNJ methods, a new method of reconstructing phylogenetic trees, FastJoin, was proposed. Experiments with sets of data showed that this new neighbor-joining algorithm yields a significant speed-up compared to classic neighbor-joining, showing empirically that FastJoin is superior to almost all other neighbor-joining implementations.

Key words: Phylogenetic tree; Neighbor-joining algorithm; FastJoin; Phylogenetic analysis

INTRODUCTION

Evolutionary trees are the basic tools for analysing differences between species, so phylogenetic analysis is a critical step in inferring the evolution of species and comprehending the information of genes and proteins. In order to better understand evolution of species, a precise method of reconstructing phylogenetic trees seems very important.

Neighbor-joining (Saitou and Nei, 1987; improved by Studier and Keppler, 1988) based on the minimum evolution principle is a widely used method for constructing phylogenetic trees, due to its elegance and speed (Nakhleh et al., 2002). Neighbor-joining is a greedy algorithm, which endeavors to minimize the sum of all branch lengths of the reconstructed tree. It requires a distance matrix $D = (D_{ij})_{nxn}$ as input, where D_{ij} is the observed distance between taxa i and j, which is computed by some evolution models. More specifically, neighbor-joining method begins with a star tree, then iteratively picks two nodes adjacent to the root and then joins them, by inserting a new node between the root and the two selected nodes. Among all possible pairs of nodes, the pair of nodes, corresponding to the minimum sum computed by Equation 1, is merged as one node.

$$S_{ij} = \frac{1}{2(N-2)} \sum_{k \neq i,j}^{N} (D_{ik} + D_{jk}) + \frac{1}{2} D_{ij} + \frac{1}{N-2} \sum_{1 \le s < t < N, s,t \neq i,j} D_{st} \quad \text{(Equation 1)}$$

Each iteration needs to consider $O(n^2)$ possible joins, and the whole neighbor-joining algorithm needs to iterate n times approximately, so neighbor-joining method requires time in $O(n^3)$ and space in $O(n^2)$ to reconstruct phylogenetic trees, where n is the number of taxa.

Here some related studies are listed. QuickTree (Howe et al., 2002) is an efficient implementation of the canonical neighbor-joining algorithm by making a pre-processing to the identical sequences. It increases the speed by a factor in time compared with neighbor-joining method. Accordingly it costs $O(n^3)$ time. QuickJoin (Mailund and Pedersen, 2004; Mailund et al., 2006), RapidNJ (Simonsen et al., 2008), NINJA (Wheeler, 2009), and the acceleration of neighbor-joining algorithm (Zaslavsky and Tatusova, 2008) all produce the same phylogenetic trees as the neighbor-joining method and improve running time by using some techniques to find the globally minimum value of sum matrix rather than by traversing the whole sum matrix in each iteration. Although all these methods are $O(n^3)$ running time in the worst case, in practice they do not need so much time. The ERapidNJ method (Simonsen et al., 2011) improves the performance of RapidNJ algorithm by means of the external memory to reduce the memory requirements of RapidNJ for reconstructing phylogenetic trees of the large data sets. Relaxed neighbor-joining (RNJ) (Evans et al., 2006; Sheneman et al., 2006) and fast neighbor-joining (Elias and Lagergren, 2005, 2009) modify the selection criteria. FastTree (Price et al., 2009, 2010) improves not only the performance of neighbor-joining method, but also the computation time and memory time of distance matrix.

In this paper, we present a theoretically improved version of canonical neighbor-joining that joins two pairs of nodes with the global minimum sum and the second minimum sum to merge as corresponding to two new nodes in each iteration instead of joining one pair of nodes with the global minimum sum for one iteration of canonical neighbor-joining, which lowers the computing time of canonical neighbor-joining and is proven in theory. And then combine this improved neighbor-joining algorithm with styles upper bound computation optimization of RapidNJ and external storage of ERapidNJ to create a new method of reconstructing phylogenetic trees - FastJoin. We evaluate the performance of FastJoin by comparing running times of an implementation of the FastJoin method with other implementations for building canonical neighbor-joining trees.

METHODS

Canonical neighbor-joining

For simplicity we use the term nodes to describe the inserted terms and taxa. Neighbor-joining is a hierarchical clustering algorithm. It takes a distance matrix $D = (D_{ij})_{nxn}$ as input, where D_{ij} is the distance between taxa i and j. Taxa are then iteratively joined using a greedy algorithm, which minimizes the total sum of branch lengths in the reconstructed tree. On the whole, the algorithm needs n-3 iterations for rootless trees, where n is the number of taxa. Two taxa i and j are selected and joined into a new node in each iteration, selected by minimizing

$$Q_{ij} = (r-2)D_{ij} - u(i) - u(j) \qquad \text{(Equation 2)}$$

where

$$u(l) = \sum_k D_{lk} \qquad \text{(Equation 3)}$$

And r is the number of unresolved nodes. Equation 2 is the deformation of Equation 1. When the minimum Q-value

$$q_1 = \min\left\{Q_{ij}\ |_{0\le i,j\le r}\right\} = Q_{i_0 j_0} \qquad \text{(Equation 4)}$$

is found, D is updated, by removing the i_0'th and j_0'th rows and columns and inserting a new row and a new column distances between the new node and the unresolved taxa, and then r reduces by one. This-operation continues until $r \le 3$. Distance between the new node formed by joined the two taxa i_0 and j_0, denoted by $a = i_0 \cup j_0$, and an old taxon l, is calculated by

$$D_{al} = \left(D_{i_0 l} + D_{j_0 l} - D_{i_0 j_0} \right) \Big/ 2 \qquad \text{(Equation 5)}$$

Each iteration of the canonical neighbor-joining algorithm takes time $O(r^2)$ to traverse all of Q. Therefore, it needs $O(n^3)$ time and $O(n^2)$ space. The result of the algorithm is an unrooted bifurcating tree where the initial taxa correspond to leaves and each joining corresponds to inserting an internal node in the tree.

RapidNJ and ERapidNJ

RapidNJ computers an upper bound on the values of Q, which is used to decrease the search scope when searching for a minimum Q-value. For utilizing the upper bound two new matrixes, S and I, are needed. S is a sorted representation of D with each row in increasing order and I maps S to D. Memory consumption is increased due to S and I, which limits inference of large phylogenetic trees although it reduces the running time of the neighbor-joining. ERapidNJ is the extension for RapidNJ that reduces the memory requirements of RapidNJ and allows phylogenies with more than 50,000 taxa to be inferred efficiently on a desktop computer. Both RapidNJ and ERapidNJ reconstruct phylogeny trees by identifying and joining the pair of nodes with global minimum transformed distance.

Improved neighbor-joining algorithm

The neighbor-joining method has proven that for a purely additive tree, i.e., its distance matrix D is purely additive, taxa i and j are true neighbors when Q_{ij} is smallest among all Q_{st}'s. According to this theory, the neighbor-joining method was set up. Here we can prove that the smallest value

$$q = \min\left\{ \left\{Q_{st} \,|_{0\le s,t\le r}\right\} \setminus \left\{Q_{i_0 l} \,|_{0\le l\le r}\right\} \setminus \left\{Q_{l i_0} \,|_{0\le l\le r}\right\} \setminus \left\{Q_{j_0 l} \,|_{0\le l\le r}\right\} \setminus \left\{Q_{l j_0} \,|_{0\le l\le r}\right\} \right\} = Q_{s_0 t_0} \qquad \text{(Equation 6)}$$

That is, the smallest value of the remaining members of Q removed the i_0'th and j_0'th row and column members, also gives the true neighbors when the distance matrix D is purely additive, where $Q_{i_0 j_0}$ is the smallest Q-value.

Saitou and Nei (1987) have proven this lemma: if i and j are true neighbors, then Q_{ij} is the minimum value in their rows and columns. Following is the description and proof of a new theorem. In the following theorem and the whole process of proving, we let the distance matrix D be purely additive.

Theorem: If i and j are chosen by the Equation 6, then i and j are true neighbors.

Proof: In the whole process of proving, we suppose that the smallest Q-value is Q_{12} and the smallest value of the remaining members of Q removed the 1'th and 2'th row and column is Q_{ij} for simplification. So 1 and 2 are true neighbors proven by Studier and Keppler, here $i,j \neq 1,2$. Now we prove that i and j are true neighbors.

When $n \leq 4$, this conclusion is obviously tenable for any purely additive tree. At the moment the only remaining element in Q except the 1'th and 2'th row and column elements is Q_{34} when $n = 4$. So Q_{34} is the smallest value of the remaining elements in Q. Three and 4 are true neighbors because 1 and 2 are true neighbors.

When $n \geq 5$, suppose that i and j are not neighbors. Neither i nor j has a neighbor according to the above lemma. At least it contains another pair of neighbors except the pair of 1 and 2 for the cases of more than four taxa for any tree. Let k and l be another pair of neighbors, so that i, j, k, and l are distinct and are represented by the tree in Figure 1. From the definition of Q_{ij}, the following Equation can be obtained by

$$Q_{kl} - Q_{ij} = \sum_{m \neq i,j,k,l} \left[\left(D_{im} + D_{jm} - D_{ij} \right) - \left(D_{km} + D_{lm} - D_{kl} \right) \right] \quad \text{(Equation 7)}$$

Obviously the Equation 7 is non-negative because D is purely additive matrix. If m is a fifth taxon, then it joins the tree in Figure 1 at a point x along one of the labeled arcs. Call that m is of type 1 if it joins the path from i to j at any node and that m is of type 2 if it joins the path from u to v at any node.

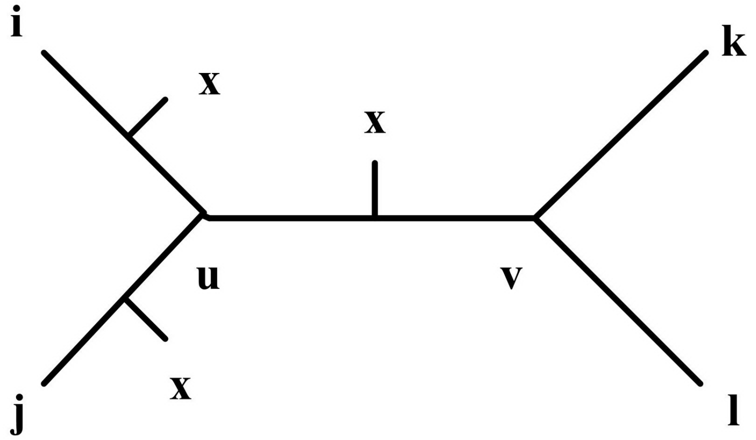

Figure 1. For the distinct taxa i, j, k, and l, the subtree includes two internal nodes u and v. k and l are neighbors, the x's represent locations at which other taxa can intercept this subtree.

In one case, if m is of type 1, then the corresponding sum in Equation 7 is $-2D_{xu}-2D_{uv}$, and in the other, if m is of type 2, then the corresponding sum in Equation 5 is $2D_{uv}-4D_{xv}$. Because the Equation 7 is non-negative, the terms corresponding to taxa m of type 2 are not less than the terms corresponding to taxa m of type 1. It shows that there are more taxa that join the path from i to j than the taxa that join the path from u to v. We have explained that neither i nor j has a neighbor, so there must be a pair of neighbors, which are s node and t node linked by o that intersect the path from i to j at some node p, which is different from u. Figure 2 illustrates relationships of these nodes, which is a part of a subtree. The above theory is applied to p, so there are more taxa that join the path from p to o than the taxa that join the path from i to j. It follows that the consequent about u and p contradict each other, so our assertion is also true for $n \geq 5$. Accordingly this theorem follows.

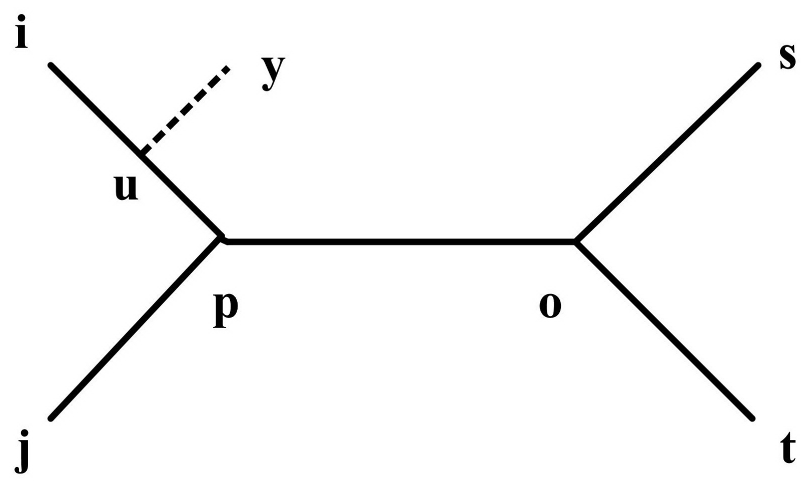

Figure 2. For the distinct taxa i, j, s, and t, the branch y stands for a subtree including the taxa k and l.

Based on the above theorem, the improved neighbor-joining algorithm is set up. In improved neighbor-joining method, the smallest value of Q, i.e., $q_1 = Q_{i_0 j_0}$, and the smallest value of remaining members of Q removed the i_0'th and j_0'th rows and columns, i.e., $q_2 = Q_{s_0 t_0}$, are found, and then D will be updated by removing the i_0'th, j_0'th, s_0'th, and t_0'th row and column. Two new rows and columns are inserted into distance matrix, which are distances between two new nodes and the old nodes, and then the number of remaining members r reduces by two. Distances between new nodes and old nodes can be calculated by Equation 5, where new nodes are $a = i_0 \cup j_0$ and $b = s_0 \cup t_0$. Distance between two new nodes a and b is calculated by

$$D_{ab} = \left(D_{i_0 s_0} + D_{j_0 s_0} + D_{i_0 t_0} + D_{j_0 t_0} - D_{i_0 j_0} - D_{s_0 t_0} \right) \big/ 4 \qquad \text{(Equation 8)}$$

These operations continue until $r \leq 3$.

It is pointed out that in practice, even these pairs may not be pairs of true neighbors; but, for a purely additive tree with no backward and parallel substitutions, this method is known to choose pairs of true neighbors.

In order to better describe the improved neighbor-joining algorithm and the neighbor-joining algorithm, we show an example of the paper of Saitou and Nei (1987), which true topology is described by Figure 3, with eight taxa 1-8, and Table 1 is the lower triangular distance matrix between any two taxa inferred by this true tree. Figure 4 and Figure 5 illustrate results of application of the canonical neighbor-joining method and the improved neighbor-joining method, respectively. Until $r \leq 3$, r standing for the number of unresolved nodes, neighbor-joining executes six times, while the improved neighbor-joining just needs three times, and the reconstructed phylogenetic trees of the two methods are exactly the same finally. For n taxa, the neighbor-joining needs n - 3 times iterations until $r \leq 3$. But, in contrast, the improved neighbor-joining algorithm just needs (n - 2) / 2 times iterations for even n and (n - 3) / 2 times iterations for odd n until $r \leq 3$.

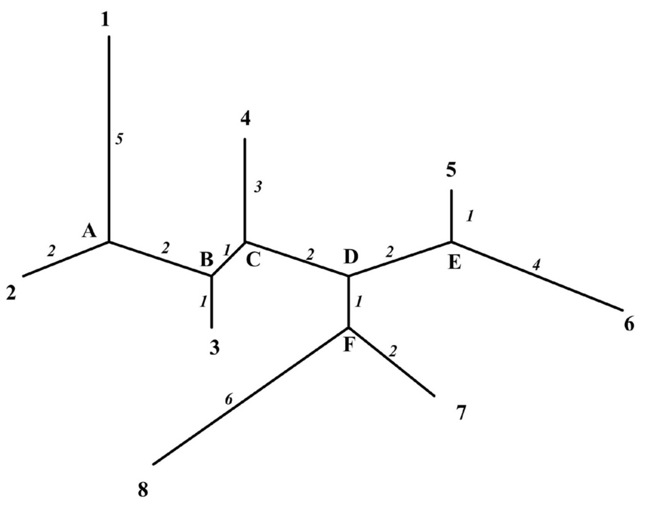

Figure 3. An unrooted tree of eight taxa, 1-8; A-F are interior nodes, and italic numbers are branch lengths of tree.

Table 1. Lower triangular distance matrix for the tree in Figure 3.

Taxa	Taxa						
	1	2	3	4	5	6	7
2	7						
3	8	5					
4	11	8	5				
5	13	10	7	8			
6	16	13	10	11	5		
7	13	10	7	8	6	9	
8	17	14	11	12	10	13	8

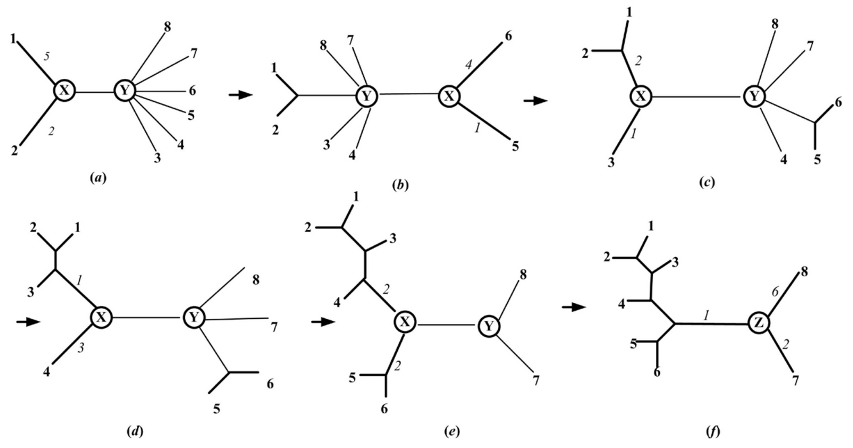

Figure 4. Application of the neighbor-joining method in the distance matrix of Table 1. Blackbody numbers are taxa, which need to cluster. Italic numbers are branch lengths, and branches with thicker lines indicate that their lengths have been determined. X and Y are the inserted new node and the center node, respectively. Z stands for the remaining internal node.

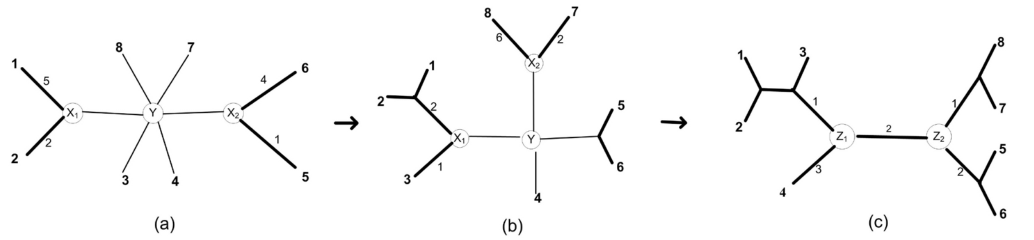

Figure 5. Application of the improved neighbor-joining algorithm to the distance matrix of Table 1. The notes are the same as that of Figure 4.

The detail procedures of the neighbor-joining algorithm and the improved neighbor-joining algorithm are listed as follows:

```
Neighbor-joining algorithm
Input: a distance matrix $D = (D_{ij})_{nxn}$
Output: a phylogenetic tree
    1. Compute the sum matrix $Q = \left(Q_{ij}\right)_{n\times n}$, according to the Equations 1 and 2
    2. Find the smallest value from matrix $Q$, which value supposed is $Q_{i_0 j_0}$, and then
         take $i_0$ and $j_0$ as a neighbor, which is denoted as $a = i_0 \cup j_0$
    3. $n = n$-1
    4. Update distance matrix $D$ by Equation 5
    5. If ($n > 3$) go to 1
```

```
Improved neighbor-joining algorithm
Input: a distance matrix $D = (D_{ij})_{nxn}$
Output: a phylogenetic tree
    1. Compute the matrix $Q = \left(Q_{ij}\right)_{n\times n}$, according to Equations 2 and 3
    2. Find out the smallest $Q$-value $Q_{i_0 j_0}$, and then take $i_0$ and $j_0$ as a neighbor, which
         is denoted as $a$
    3. Find out the smallest value from Equation 6, which is $Q_{s_0 t_0}$, and then take $s_0$ and
         $t_0$ as a neighbor, which is denoted as $b = s_0 \cup t_0$
    4. $n = n$-2
    5. Update distance matrix $D$ by Equations 5 and 8
    6. If ($n > 3$) go to 1
```

The similarity between improved neighbor-joining algorithm and neighbor-joining algorithm is that they search true neighbors at each iteration if their true tree is a purely additive tree. So the reconstructed trees by improved neighbor-joining algorithm and by neighbor-joining algorithm are identical. From the descriptions of the above algorithms, it is obvious that the main difference between two methods is that the improved algorithm adds a step on the basis of neighbor-joining algorithm in which step we find out that the neighbors can be found out in the another step of neighbor-joining, and the needed time of updating distance matrix of the improved algorithm is almost half of neighbor-joining. The search time of neighbor-joining algorithm is $n^2+(n\text{-}1)^2+(n\text{-}2)^2+(n\text{-}3)^2+\ldots+5^2+4^2$, while the search time of the improved algorithm is $n^2+(n\text{-}2)^2+(n\text{-}2)^2+(n\text{-}4)^2+\ldots+6^2+4^2$ and the update time of the improved neighbor-joining algorithm is also reduced. The running time of canonical neighbor-joining method comes mainly from the search time for the pair of nodes to join and the update time of distance matrix after joining. The improved neighbor-joining algorithm reduces running time of neighbor-joining algorithm by decreasing the search time and the updating time of distance matrix. So the neighbor-joining algorithm costs $n^3/3+O(n^2)$ time and the improved neighbor-joining algorithm costs $n^3/6+O(n^2)$ time. The improved neighbor-joining algorithm costs $O(n^3)$ time and $O(n^2)$ space the same as the neighbor-joining method. The difference is that the improved neighbor-joining algorithm increases the speed by a factor in time compared with the neighbor-joining algorithm. The most important is that the improved neighbor-joining algorithm is set up on the basis of an important theorem so that these neighbors are true neighbors if their true tree is a purely additive tree.

This improved neighbor-joining algorithm is particularly effective on data sets, which are phylip formatted distance matrices using QuickTree translated from Pfam (Finn et al., 2006) data, which gave rise to poor performance in canonical neighbor-joining (see Figure 6 and Figure 7). From the two figures, it is obvious that the improved neighbor-joining algorithm is superior to the neighbor-joining algorithm in execution time because iteration times of improved neighbor-joining are half of neighbor-joining. Although the running time of the improved neighbor-joining algorithm is not the half of neighbor-joining algorithm, which is influenced by the term $O(n^2)$ of the respective running time equation and, with the increasing of the number of taxa, the half trend is more and more obvious.

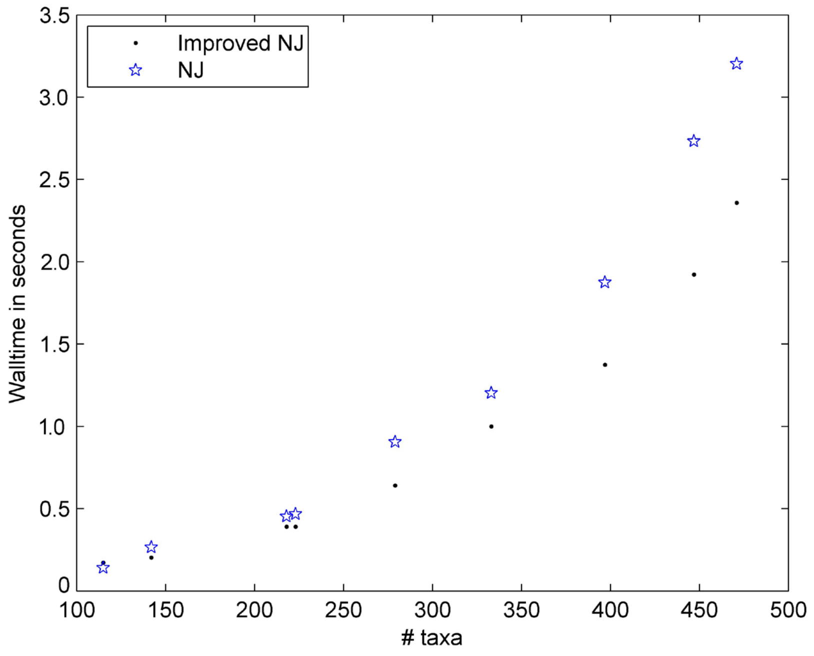

Figure 6. Performance of improved neighbor-joining (NJ) algorithm compared to neighbor-joining on small Pfam data (<500 taxa).

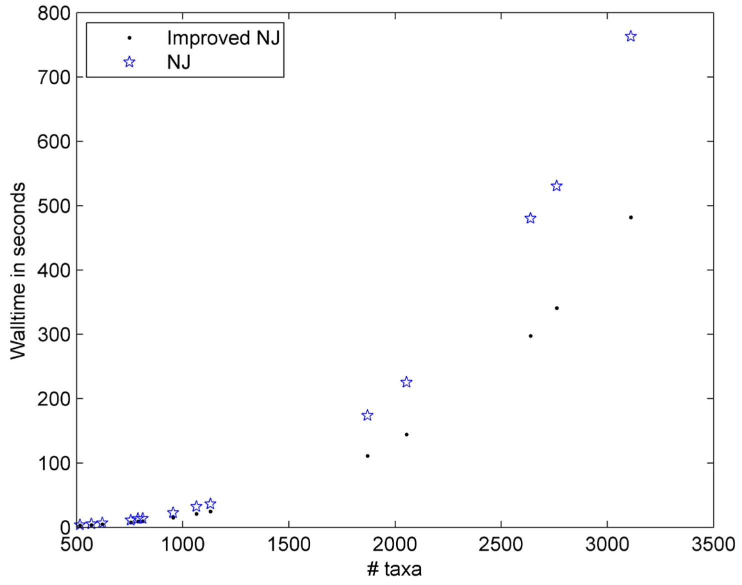

Figure 7. Performance of improved neighbor-joining (NJ) algorithm compared to neighbor-joining on large Pfam data (500-3500 taxa).

The improved neighbor-joining algorithm reduces the running time of canonical neighbor-joining by increasing the speed of joining, which search strategy can choose any advanced one of neighbor-joining. Consequently we choose, to the best of our knowledge, the fastest search strategy of RapidNJ and the external storage of ERapidNJ for large scale data, and then combine it with the improved neighbor-joining algorithm to create a new advanced method of neighbor-joining, FastJoin, for reducing the running time of reconstructing phylogenetic trees. The results of experiments show that the running time of FastJoin is reduced on the majority of experimental data.

FastJoin

By combining the improved neighbor-joining algorithm with the upper bound computation optimizations of RapidNJ and the external storage style of ERapidNJ methods, a new method of reconstructing phylogenetic trees, we create a new algorithm - FastJoin, which finds two pairs of nodes to merge as corresponding to two new nodes and then updates the distance matrix in each iteration instead of finding a pair of nodes of RapidNJ and ERapidNJ. The search strategy of FastJoin is the same as that of RapidNJ and the external memory of FastJoin is the same as that of ERapidNJ.

RESULTS

To evaluate the performance of FastJoin, we compared running times with RapidNJ, ERapidNJ and some other advanced neighbor-joining on Pfam data sets. QuickJoin, Clearcut and NINJA have been used as reference in our experiments. QuickJoin is an implementation of heuristics for canonical neighbor-joining. Clearcut is an implementation for the RNJ algorithm, while NINJA uses an upper bound on *Q*-value like RapidNJ and first utilize external memory efficiently so that more than 50,000 taxa can be built in a few hours on a desktop computer with only 2 GB of RAM.

QuickJoin, RapidNJ and ERapidNJ are implemented in C++, Clearcut in C while NINJA is implemented in Java.

QuickTree, which is a fast implementation with a heuristic for treating identical sequences to reduce the running time of canonical neighbor-joining, is not included in our experiments. In Simonsen et al. (2008, 2011), there are the comparisons of the RapidNJ with QuickTree and ERapidNJ with QuickTree. There are three programs using external memory in order to reconstruct trees of large scale datasets.

All experiments were performed on machines with an Intel Xeon E5504 2.0 GHz CPU, 8 GB RAM and 147 GB HDD. The operating system was Debian 4.1 32 bit with Java 1.6 installed. FastJoin was written in C++. We used the "real-time" output from the standard time tool for measurements of running time. Five runs of each program on every data set were made, and the best time of the five runs was used in order to avoid perturbation from other processes on the system.

All implementations need a distance matrice in Phylip format as input. The data used are protein sequence alignments from Pfam, which are translated into Phylip formatted distance matrices using Quick-Tree as input to all tools. Pfam data sets represent real-data and can test how efficient the different algorithms used in the implementations. Figure 8 and Figure 9 show the results of the experiments on these data, which indicate that FastJoin is faster than all internal memory advanced neighbor-joining and is faster than many external memory advanced neighbor-joining on large data sets.

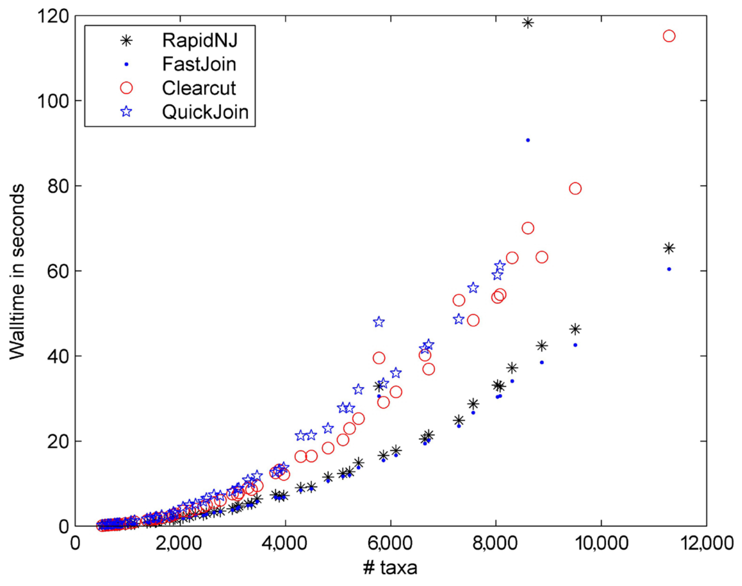

Figure 8. Performance of FastJoin compared to RapidNJ, Clearcut and QuickJoin on data sets with 500 to 12,000.

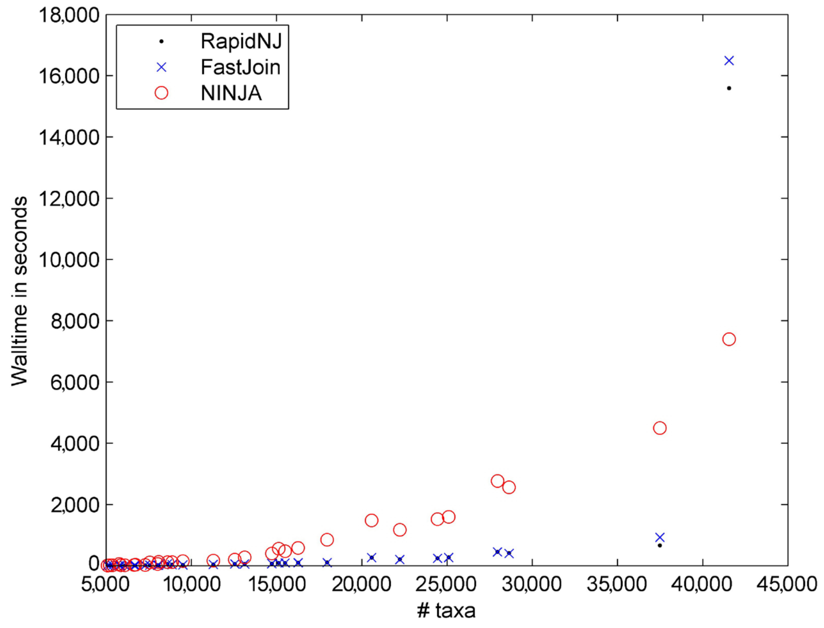

Figure 9. Performance of FastJoin compared to ERapidNJ and NINJA on data sets with 5000 to 45,000.

DISCUSSION

Figure 8 shows the implementations of four internal memory versions of advanced neighbor-joining. QuickJoin has the longest running time, while FastJoin has the shortest, which is faster than the tool RapidNJ, and with the increasing of the data scale, the saving time of the FastJoin algorithm for reconstructing phylogenetic trees is more and more. Some outliers can be observed in Figure 8, where the running time of FastJoin is more than the running time of Clearcut, QuickJoin and RapidNJ, only less than the running time of NINJA. The FastJoin algorithm is extension of RapidNJ and ERapidNJ by applying the improved canonical neighbor-joining algorithm. In Figure 8 these outliers are just the outliers of RapidNJ method, where the running time of RapidNJ is more than the running time of Clearcut and QuickJoin and less than NINJA, so the reason of these outliers is the same as the reason of RapidNJ, which is explained specifically in Simonsen et al. (2008).

Figure 9 shows the implementations of three external memory versions of advanced neighbor-joining on data sets with 5000 to 45,000 taxa, while larger data sets cannot be operated subject to the restriction of the size of HDD. As seen in Figure 9 the FastJoin method is faster than the NINJA method on many data sets and has almost the same speed of reconstructing phylogenetic trees with the ERapidNJ method. Except for a few outliers from NINJA, FastJoin and ERapidNJ have roughly the same running time and both of them have less running time than NINJA, which is consentaneous with the results of paper of Simonsen et al. (2011). It is not obvious that the FastJoin algorithm is superior to the ERapidNJ algorithm from Figure 9 since the running time of FastJoin algorithm is a little less than that of the ERapidNJ algorithm except some outliers. The FastJoin algorithm is superior to the NINJA algorithm except some outliers. There are several outliers, where the running time of FastJoin is slightly more than that of ERapidNJ and FastJoin and ERapidNJ have more running time than NINJA has because NINJA computers a tighter bound on Q-values than FastJoin and ERapidNJ. Hence NINJA has an advantage over FastJoin and ERapidNJ for some large scale data sets. A result with 46,279 taxa is not shown in Figure 9, because of the lack of HDD, FastJoin and ERapidNJ do not run on the machine with 100 GB HDD, while NINJA was able to finish these data within 4 h on this machine. So FastJoin and ERapidNJ need more external memory than NINJA on large data sets.

For most data sets FastJoin and ERapidNJ are superior to NINJA and FastJoin is slightly faster than ERapidNJ. If large data sets need to run and the external memory of machine is limited, I think NINJA is the best choice. Because the FastJoin method is by combining the improved neighbor-joining algorithm to the ERapidNJ method, FastJoin and ERapidNJ have almost identical properties. The real running time of the FastJoin method is slightly less than the running time of ERapidNJ, which is not obvious on figure.

CONCLUSIONS

Experiment results and analysis of time complexity of algorithm showed that the improved neighbor-joining algorithm is superior to the canonical neighbor-joining algorithm. By combining this improved algorithm with the RapidNJ and the ERapidNJ method, we created the FastJoin method. The FastJoin method improves the running time by reducing the scale of searching and the time of joining. While the worst case running time of FastJoin is still $O(n^3)$, it is superior to many advanced neighbor-joining such as RapidNJ, Clearcut, QuickJoin. When using the external memory, the FastJoin method exceeds NINJA on majority data sets and FastJoin outperforms ERapidNJ slightly on many data sets. The results of this experiment tell us that FastJoin and ERapidNJ have more external memory compared with NINJA.

ACKNOWLEDGMENTS

Research supported by the Natural Science Foundation of China (under grant #60932008, #61172098), the Specialized Research Fund for the Doctoral Program of Higher Education of China (under grant #20112302110040) and the Fundamental Research Funds for the Central Universities (under grant #HIT.ICRST.2010022).

REFERENCES

Elias I and Lagergren J (2005). Fast Neighbor Joining. In: Proceedings of the 32nd International Colloquium: 11-15 July 2005 (L Caires, GF Italiano, L Monteiro, C Palamidessi, et al., eds.). Springer Berlin Heidelberg, Lisbon, 1263-1274.

Elias I and Lagergren J (2009). Fast neighbor joining. *Theor. Comput.* 410: 1993-2000.

Evans J, Sheneman L and Foster J (2006). Relaxed neighbor joining: a fast distance-based phylogenetic tree construction method. *J. Mol. Evol.* 62: 785-792.

Finn RD, Mistry J, Schuster-Bockler B, Griffiths-Jones S, et al. (2006). Pfam: clans, web tools and services. *Nucleic Acids Res.* 34: D247-D251.

Howe K, Bateman A and Durbin R (2002). QuickTree: building huge neighbor-joining trees of protein sequences. *Bioinformatics* 18: 1546-1547.

Mailund T and Pedersen CNS (2004). QuickJoin - fast neighbor-joining tree reconstruction. *Bioinformatics* 20: 3261-3262.

Mailund T, Brodal GS, Fagerberg R, Pedersen CNS, et al. (2006). Recrafting the neighbor-joining method. *BMC Bioinformatics* 7: 29.

Nakhleh L, Moret BME, Roshan U and John KS (2002). The Accuracy of Fast Phylogenetic Methods for Large Datasets. In: Proceedings of the Seventh Pacific Symposium on Biocomputing: 2001 (Altman RB, Dunker AK, Hunter L, Lauderdale K, et al., eds.). World Scientific, Singapore, 211-222.

Price MN, Dehal PS and Arkin AP (2009). FastTree: computing large minimum evolution trees with profiles instead of a distance matrix. *Mol. Biol. Evol.* 26: 1641-1650.

Price MN, Dehal PS and Arkin AP (2010). FastTree 2-approximately maximum-likelihood trees for large alignments. *PLoS One* 5: e9490.

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

Sheneman L, Evans J and Foster JA (2006). Clearcut: a fast implementation of relaxed neighbor joining. *Bioinformatics* 22: 2823-2824.

Simonsen M, Mailund T and Pedersen CNS (2008). Rapid Neighbor-Joining. In: Proceedings of the Eighth International Workshop on Algorithms in Bioinformatics: 15-19 September 2008 (Crandall KA and Lagergren J, eds.). Springer Berlin Heidelberg, Karlsruhe, 113-122.

Simonsen M, Mailund T and Pedersen CNS (2011). Inference of Large Phylogenies Using Neighbor-Joining. In: Proceedings of the Third International Joint Conference: 20-23 January 2010 (Fred A, Filipe J and Gamboa H, eds.). Springer Berlin Heidelberg, Valencia, 334-344.

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

Wheeler TJ (2009). Large-Scale Neighbor-Joining with NINJA. In: Proceedings of the Ninth International Workshop on Algorithms in Bioinformatics: 12-13 September 2009 (Salzberg SL and Warnow T, eds.). Springer Berlin Heidelberg, Philadelphia, 375-389.

Zaslavsky L and Tatusova T (2008). Accelerating the Neighbor-Joining Algorithm Using the Adaptive Bucket Data Structure. In: Proceedings of the Fourth International Symposium: 6-9 May 2008 (Mandoiu I, Sunderraman R and Zelikovsky A, eds.). Springer Berlin Heidelberg, Atlanta, 122-133.