Received August 19, 2019, accepted September 1, 2019, date of publication September 9, 2019, date of current version September 23, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2939952

Analysis on Algorithms for Constructing Phylogenetic Trees From Distances

JUAN WANG
School of Computer Science, Inner Mongolia University, Hohhot 010021, China
e-mail: wangjuan@imu.edu.cn

This work was supported by the National Natural Science Foundation of China under Grant 61661040 and Grant 61661039.

ABSTRACT Neighbour-joining algorithm (NJ for short) is an used widely algorithm for constructing phylogenetic trees from distances because of its high accuracy. For NJ costs a lot of time to construct phylogenetic trees for the large input data, it does not often output a result within feasible time. Until now, there are several improved algorithms of NJ aiming for speeding up the construction of trees, but there is no research on the accuracy of those improved algorithms. The paper will analyze and compare the accuracy of NJ as well as its improved algorithms by means of the experiments. We introduce a new improved algorithm, called RandomNJ, which is an efficient method for constructing phylogenetic trees from distances. Furthermore, we design the INJ which is a web-based server for on-line constructing the phylogenetic trees using the improved algorithms and NJ. It is available from http://bioinformatics.imu.edu.cn/INJ/.

INDEX TERMS Phylogenetic trees, neighbour-joining, NJ, distances.

I. INTRODUCTION

NJ is initially proposed by Saitou and Nei [19] which can construct a phylogenetic tree. Its input is a distance matrix in which one value describes the evolutionary dissimilarity between a pair of taxa. And then Studier and Keppler improve NJ by means of slightly changing the key computation formula in NJ algorithm so that the running time of NJ can be computed [23]. It needs $O(n^3)$ running time and $O(n^2)$ space consumption, where n is the number of taxa. NJ has a wide utilization in the bioinformatics [2], [13]. NJMerge and TreeMerge are two modification of NJ in order to estimate the species tree based on a divide-and-conquer framework [16], [17]. NJ is also used to predict the protein function. Reference [10] first uses NJ method to construct a phylogenetic tree for a protein set and then uses the phylogenetic distances between proteins to predict the protein function. In order to better construct phylogenetic trees for the increasing amount of available data, many scientists begin to improve the NJ algorithm, and obtain a lot of improved algorithms. NJ has an important theoretical evidence, i.e. it can construct right trees for the additive distance matrices (will be introduced in the following). NJ needs to seek the minimum value from a matrix in each iteration of construction. In order to speed up the construction of phylogenetic trees, the improved algorithms try to change the search strategy of the minimum value in the iteration. So there are several improved methods that can not construct a right tree even for additive distance matrices. Here all improved methods can be divided into two classes based on the constructed trees for the additive distance matrices.

One class is the algorithms which have the same theoretical evidence as NJ and can construct a right tree for additive distance matrices. QuickJoin uses heuristics to speed up the NJ by building a quad-tree to prune the search scope of minimum [14]. RapidNJ reduces the scope of search by calculating an upper bound when searching for the minimum value [21]. RapidNJ needs to add two new matrices for excluding a lot of members, so its consumption of memory is increased. ERapidNJ is the improvement of RapidNJ in order to reduce the internal memory requirements of RapidNJ by means of external memory [22], [26], [30]. FastJoin searches two minimum values in each iteration aiming to reduce the running time of NJ [24].

The other class is the algorithms which do not have any theoretical evidence and do not construct right trees for the additive distance matrices. BIONJ [7], [8] computes the new distance matrix by a first-order covariance matrix of the distance matrix in each iteration, aiming to construct a more practical phylogenetic tree for species. GNJ uses a sampling of the solution space obtained by keeping track of multiple partial solutions during its execution [18]. Clearcut searches

The associate editor coordinating the review of this manuscript and approving it for publication was Vincenzo Conti.

a local minimum value when constructing phylogenetic trees [6], [20]. Fast neighbour-joining (FNJ for short) method is based on a auxiliary function to find a local minimum in order to decrease the running time of NJ [4], [5], [11].

Up to now, several literatures have studied the efficiency of NJ method. References [1], [12], [15] have studied the efficiency of NJ on reconstructing deep and shallow evolutionary relationships for large phylogenies. And it follows that it is more difficult to reconstruct the deep branches than the shallower branches using NJ method. The taxon ordering of phylogenetic trees plays an important role in the construction of trees when using the NJ [3], [9].

Almost all of the improved algorithms of NJ aim to speed up the construction of trees, and their running times are compared in many papers [24], [25], [29]. However, there is not any one research on their accuracy of the phylogenetic trees constructed by those improved algorithms and NJ. Here we research on the accuracy of several typically improved algorithms, including NJ, FastJoin, RapidNJ, Clearcut, when they construct phylogenetic trees. NJ needs to search a minimum value in each iteration. It is possible that there are several minimum values in some iteration, while those improved algorithms do not give a reasonable approach on the situation. Here we propose a new improved algorithm of NJ, called as RandomNJ, by randomly choose a minimum value when there are many minimum values in the construction. Experiments show that the RandomNJ is an efficient method. Furthermore, we design a tool for researchers to conveniently construct phylogenetic trees using the NJ and its improved algorithms.

II. METHODS

For a matrix D, if there is a tree with positive branch lengths and the length of shortest path between any two leaves is the distance between the two taxa in D, then D is called additive [28]. The matrix in Table 1 is additive because the tree in Figure 1 satisfies the definition.

TABLE 1. An additive distance matrix.

taxa	1	2	3	4	5	6	7	8
1	0	7	8	11	13	16	13	17
2	7	0	5	8	10	13	10	14
3	8	5	0	5	7	10	7	11
4	11	8	5	0	8	11	8	12
5	13	10	7	8	0	5	6	10
6	16	13	10	11	5	0	9	13
7	13	10	7	8	6	9	0	8
8	17	14	11	12	10	13	8	0

NJ and its improved algorithms take a distance matrix $D_{n\times n}$ as input, and output a phylogenetic tree, where n is the number of taxa. The NJ and its improved algorithms will be introduced in detail in the following.

A. NJ

NJ is a greedy algorithm. It begins with a starlike tree (i.e. a central node O connecting n nodes). Each iteration of NJ is executed as follows. First it searches a pair of nodes which are the most potential neighbours among all possible pairs of nodes. The search is based on the following matrix S computed by the Formula 1. Then it creates a new node u to connect the selected pair of nodes and the central node, and the connection of the pair of nodes and the central node is deleted. Next the distances between u and the others node are updated and the number of taxa is updated to $n-1$. The iteration continues until $n \leq 3$.

The matrix $S = (s_{ij})_{n\times n}$ (named sum matrix) is computed from the input distance matrix D, where s_{ij} is

$$s_{ij} = (n-2)D(i,j) - R_i - R_j \quad (1 \leq i \neq j \leq n), \tag{1}$$

where $D(i,j)$ is the value in the ith row and jth column of D, and

$$R_i = \sum_{k=1}^{n} D(i,k). \tag{2}$$

The distance between the new node with other taxa is updated by the following formula,

$$D(k,u) = \frac{1}{2}(D(a,u) + D(b,u) - D(a,b)) \quad (k \neq a, b), \tag{3}$$

where s_{ab} is the minimum value from the matrix S. The two nodes a and b found by NJ are true neighbors, if the distance matrix D is additive. Algorithm 1 is the pseudo-code of NJ.

Algorithm 1 NJ

```
1: input: a distance matrix $D_{n\times n}$
2: output: a phylogenetic tree $T$
3: $T$ is a starlike tree, and $O$ is the center node of $T$;
4: while $n \geq 3$ do
5:     compute each value of $S$ by the formula 1;
6:     search a minimum value $s_{ab}$ from $S$, create a new node
       $A$ connecting $a$ and $b$, connect $A$ with $O$, and delete the
       connection between nodes $a$, $b$ and $O$;
7:     update the distance matrix $D$ by the formula 3;
8:     $n = n - 1$;
9: end while
10: return $T$;
```

The following section will introduce the improved algorithms of NJ, namely FastJoin, RapidNJ, Clearcut and RandomNJ. Among them, FastJoin, RapidNJ and RandomNJ search the minimum value in the S matrix iteratively, which is the same as NJ, so those algorithms can choose a true pair of neighbors when the input data is an additive distance matrix. While Clearcut cannot ensure that the selected pair of taxa are true neighbors because it searches a local minimum s-value each one time.

B. FASTJOIN

For an additive distance matrix, Wang et al have proved that there is another pair of true neighbors besides the true neighbors a and b corresponding to minimum value of

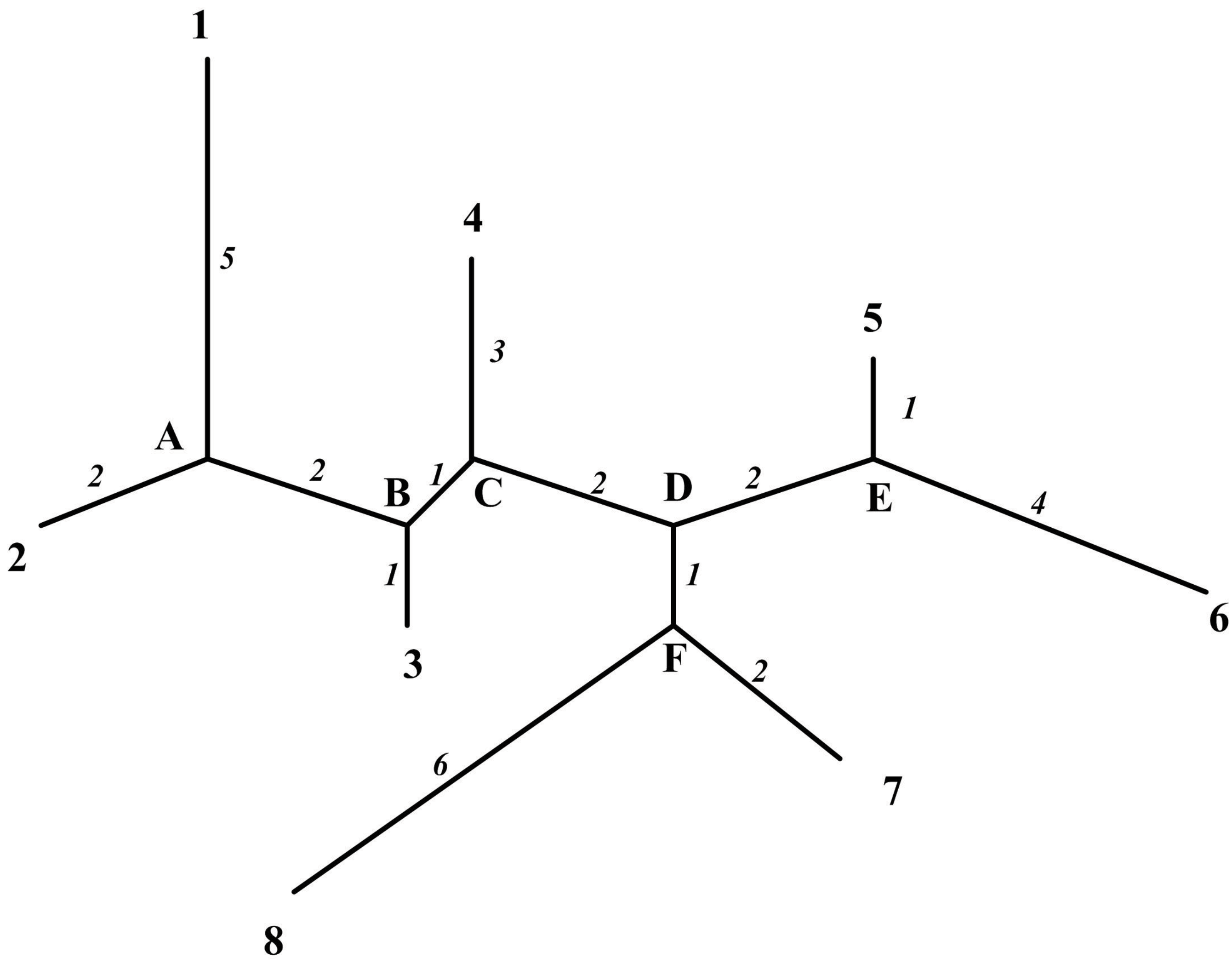


FIGURE 1. An unrooted phylogenetic tree.

matrix S [24]. The new true neighbors are the two nodes with the minimum value of the remaining members of S removed the members in ath row, ath column, bth row and bth column. FastJoin is proposed based on the theory, which is an improved algorithm for NJ by iteratively picking two pairs of nodes from S to merge as two new nodes instead of picking a pair of nodes. It saves running time for NJ algorithm.

FastJoin is executed in each iteration as follows. First it finds out the minimum value s_{ab} of S, and the minimum value s_{uv} from the rest values of S by removing the values of S in the ath row, ath column, bth row and bth column. There are two pairs of neighbors a and b as well as u and v. Then it creates a new node A connecting a, b and another new node B connecting u, v. The distances between the new node and the other nodes are computed by the formula 3. The distance between the two new nodes A and B is computed by the formula 4. Algorithm 2 is the pseudo-code of FastJoin. FastJoin has the same time complexity and space complexity as NJ, while it reduces the consumption of time by a factor in time complexity for NJ.

$$D(A,B) = \frac{1}{4}(D(a,u) + D(b,u) + D(a,v) + D(b,v) - 2 \cdot D(a,b) - 2 \cdot D(u,v)) \quad (4)$$

C. RAPIDNJ

In order to speed up the construction of trees, RapidNJ tries to decrease the search scope of S by computing an upper bound of the minimum value. It introduces two new matrices Q and I. The values in each one row of Q is the values in the same row of D sorted in ascending order, and I records the mapping from Q to D. In more detail, the values of ith row in D are sorted as $D(i, O_1) \leq D(i, O_2) \leq \cdots D(i, O_n)$, where $O_1, O_2, \cdots, O_n$ be a permutation of $1, 2, \cdots, n$. Then

$$Q(i,j) = D(i, O_j) \quad (5)$$

and

$$I(i,j) = O_j \quad (6)$$

Algorithm 2 FastJoin

```
1:  input: a distance matrix $D_{n\times n}$
2:  output: a phylogenetic tree $T$
3:  $T$ is a starlike tree, and $O$ is the center node of $T$;
4:  while $n \geq 3$ do
5:      compute each value of $S$ by the formula 1;
6:      search a minimum value $s_{ab}$ from $S$, and a minimum value $s_{uv}$ from the rest values of $S$ by removing the members in the $a$th row, $a$th column, $b$th row and $b$th column;
7:      create a new node $A$ connecting $a$ and $b$, connect $A$ with $O$, and delete the connection between nodes $a$, $b$ and $O$;
8:      create a new node $B$ connecting $u$ and $v$, connect $B$ with $O$, and delete the connection between nodes $u$, $v$ and $O$;
9:      update the distance matrix $D$ by the formulas 3 and 4;
10:     $n = n - 2$;
11: end while
12: return $T$;
```

For $1 \leq l \leq n$, the $u(l)$ is computed by the following formula, and let U_{max} be the maximum value in all $u(l)$.

$$u(l) = \sum_{k=1}^{n} D(l,k)/(n-2) \tag{7}$$

The upper bound of S is computed by the following method.

1. Let $s_{min} = \infty, i = -1, j = -1$;
2. For $Q(r,c)$
   a. if $Q(r,c) - u(r) - U_{max} > s_{min}$, then go to the next row;
   b. if $S(r, I(r,c)) < s_{min}$, then $s_{min} = S(r, I(r,c)), i = r, j = I(r,c)$

If the formula 8 is true, then the $Q(r,k)(c \leq k \leq n)$ is so large that the $S(r, I(r,k))$ is more than the current s_{min}, so rest values in the rth row of Q are impossible to be the minimum and are stopped to search. RapidNJ is by visiting values in the row of Q to search the minimum value in S, where the searched values in each row of Q depend on the present s_{min}. If the present s_{min} is close to the minimum value of S, then a lot of values will be eliminated from S in the following search. The RapidNJ reduces the running time of NJ in the practical application. In the worse case, the RapidNJ will search each value of each row in Q, so the worst case time complexity of RapidNJ is $O(n^3)$.

$$Q(r,c) - u(r) - U_{max} > s_{min} \tag{8}$$

D. CLEARCUT

Clearcut searches a local minimum value rather than a global minimum of S, so it is also called relaxed neighbor joining (RNJ for short) [6]. For a taxon a, it first computes E_a by the following formula:

$$E_a = \{E_{ai} | 1 \leq i \leq n, i \neq a\}, \tag{9}$$

where E_{ai} is

$$E_{ai} = D(a,i) - (\sum_{k=1,k\neq i}^{n} D(a,i) + \sum_{j=1,j\neq a}^{n} D(i,j))/(n-2) \tag{10}$$

In each one iteration, Clearcut picks randomly two taxon a, b, if E_{ab} is the minimum value of E_a and the minimum value of E_b, then it takes a, b as a neighbour and connects them with a new node. Algorithm 3 is the pseudo-code of Clearcut. The time complexity of Clearcut is $O(n^2 logn)$. The two taxa selected by the Clearcut are not necessarily true neighbours for an additive distance matrix as input.

Algorithm 3 Clearcut

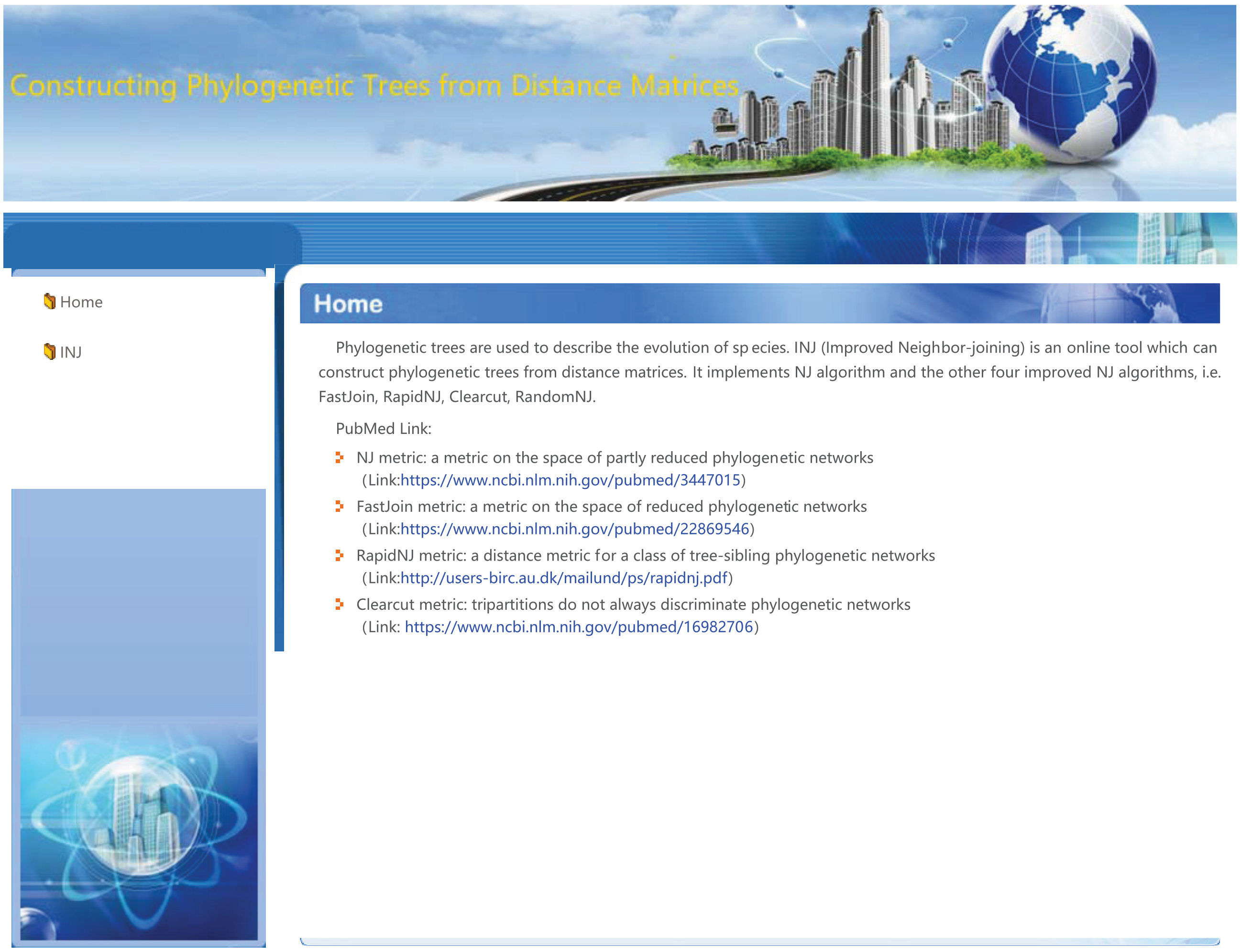
```
1:  input: a distance matrix $D_{n\times n}$
2:  output: a phylogenetic tree $T$
3:  $T$ is a starlike tree, and $O$ is the center node of $T$;
4:  while $n \geq 3$ do
5:      randomly choose two different taxa $a$ and $b$;
6:      compute the set $E_a = \{E_{ai} | 1 \leq i \leq n$ and $i \neq a\}$;
7:      compute the set $E_b = \{E_{bj} | 1 \leq j \leq n$ and $j \neq b\}$;
8:      if $E_{ab}$ is the minimum number of $E_a$ and the minimum number of $E_b$ then
9:          create a new node $B$ connecting $a$ and $b$, connect $B$ with $O$, and delete the connection between nodes $a$, $b$ and $O$;
10:     end if
11:     update the distance matrix $D$ by the formula 3;
12:     $n = n - 1$;
13: end while
14: return $T$;
```

E. RANDOMNJ

The above algorithms need to search a minimum number and then merge the two taxa in its each iteration. If there are more than two minimum values in one iteration, then the constructed tree depends on the first minimum value obtained by algorithms. Here we introduce a new improved NJ, called RandomNJ, which randomly chooses a minimum value from all minimum values in each iteration. Algorithm 4 is the pseudo-code of RandomNJ.

III. ACCESS OF INJ

We devise the INJ (Improved NJ) in order to make it easier for the researches to construct phylogenetic trees using NJ and its improved algorithms. INJ is a web server which is used to construct phylogenetic trees. Moreover, the users can download the installable software INJ from the web server. INJ implements the above algorithms. Figure 2 shows the home page of INJ.



Phylogenetic trees are used to describe the evolution of sp ecies. INJ (Improved Neighbor-joining) is an online tool which can construct phylogenetic trees from distance matrices. It implements NJ algorithm and the other four improved NJ algorithms, i.e. FastJoin, RapidNJ, Clearcut, RandomNJ.

PubMed Link:

- NJ metric: a metric on the space of partly reduced phylogenetic networks (Link:https://www.ncbi.nlm.nih.gov/pubmed/3447015)
- FastJoin metric: a metric on the space of reduced phylogenetic networks (Link:https://www.ncbi.nlm.nih.gov/pubmed/22869546)
- RapidNJ metric: a distance metric for a class of tree-sibling phylogenetic networks (Link:http://users-birc.au.dk/mailund/ps/rapidnj.pdf)
- Clearcut metric: tripartitions do not always discriminate phylogenetic networks (Link: https://www.ncbi.nlm.nih.gov/pubmed/16982706)

FIGURE 2. The home page of INJ.

Algorithm 4 RandomNJ

```
1: input: a distance matrix $D_{n \times n}$
2: output: a phylogenetic tree $T$
3: $T$ is a starlike tree, and $O$ is the center node of $T$;
4: while $n \geq 3$ do
5:     compute each value of $S$ by the formula 1;
6:     pick randomly a minimum value $s_{ab}$ from all minimum values;
7:     create a new node $A$ connecting $a$ and $b$, connect $A$ with $O$, and delete the connection between the nodes $a$, $b$ and $O$;
8:     update the distance matrix $D$ by the formula 3;
9:     $n = n - 1$;
10: end while
11: return $T$;
```

IV. RESULTS

The experiments will use two groups of constructed data to test the performance of algorithms. First group of data is additive distance matrices, in which each one matrix is computed from all branch lengths of a phylogenetic tree. Second group of data is non-additive distance matrices, in which each one matrix is calculated from DNA sequences by means of the evolutionary model of DNA.

A. DATA

We first introduce the generation of additive distance matrices. We randomly generate 1000 unrooted phylogenetic trees with n taxa. For each generated phylogenetic tree, we set a value from 1 to 10 for each branch and compute the distance between any two taxa which is sum of all branch lengths in the shortest path from a taxa to the other taxa. For example, Table 1 shows the distance matrix of the unrooted phylogenetic tree in Figure 1.

We then introduce the generation of non-additive distance matrices. First we generate randomly a rooted phylogenetic tree P with n taxa and a DNA sequence S with length 10000 which is composed of A,G,C and T. Then we take the sequence S as the ancestor and evolve it alone the topology of P. The considered events are insert, deletion and substitution, and the evolution proportions are respectively 2%, 2% and 1%. So each leave of the tree obtains a DNA sequence

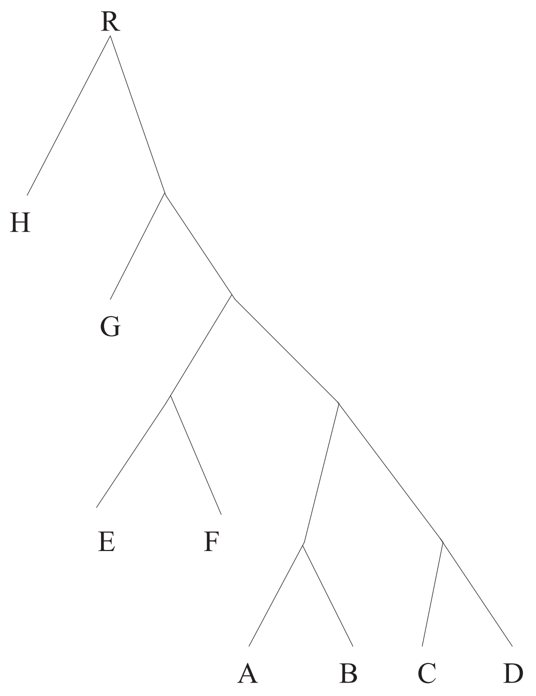


FIGURE 3. A rooted phylogenetic tree with the root node R.

TABLE 2. Results on additive distance matrices.

n	NJ	FastJoin	RapidNJ	Clearcut	RandomNJ
5	0.000	0.000	0.000	0.000	0.000
6	0.000	0.000	0.000	0.071	0.000
7	0.000	0.000	0.000	0.135	0.000
8	0.000	0.000	0.000	0.198	0.000
9	0.000	0.010	0.000	0.235	0.000
10	0.000	0.018	0.000	0.346	0.000
11	0.000	0.044	0.000	0.391	0.000
12	0.000	0.038	0.000	0.423	0.000
13	0.000	0.056	0.000	0.484	0.000
14	0.000	0.074	0.000	0.548	0.000
15	0.000	0.075	0.000	0.588	0.000
16	0.000	0.118	0.000	0.703	0.000
17	0.000	0.141	0.000	0.731	0.000
18	0.000	0.134	0.000	0.768	0.000
19	0.000	0.142	0.000	0.847	0.000
20	0.000	0.115	0.000	0.952	0.000
22	0.000	0.188	0.000	1.050	0.000
24	0.000	0.167	0.000	1.164	0.000
26	0.000	0.170	0.000	1.194	0.000
28	0.000	0.151	0.000	1.313	0.000
30	0.000	0.164	0.000	1.432	0.000
32	0.000	0.161	0.000	1.491	0.000
34	0.000	0.126	0.000	1.532	0.000
36	0.000	0.119	0.000	1.724	0.000
38	0.000	0.137	0.000	1.941	0.000
40	0.000	0.139	0.000	1.782	0.000
mean	0.000	0.0956	0.000	0.8478	0.000

TABLE 3. Results on non-additive distance matrices.

n	NJ	FastJoin	RapidNJ	Clearcut	RandomNJ
5	0.988	0.988	0.988	0.988	0.988
6	1.665	1.665	1.665	1.665	1.665
7	2.465	2.477	2.465	2.457	2.465
8	3.258	3.277	3.258	3.272	3.258
9	4.144	4.139	4.144	4.130	4.144
10	5.023	5.021	5.023	5.031	5.023
11	5.854	5.867	5.854	5.861	5.854
12	6.819	6.838	6.819	6.832	6.819
13	7.817	7.800	7.817	7.809	7.817
14	8.671	8.700	8.671	8.664	6.671
15	9.604	9.557	9.604	9.571	9.604
16	10.725	10.725	10.725	10.734	10.725
17	11.574	11.575	11.574	11.562	11.574
18	12.520	12.482	12.520	12.501	12.520
19	13.471	13.445	13.471	13.471	13.471
20	14.514	14.498	14.514	14.490	14.514
22	16.346	16.362	16.346	16.368	16.346
24	18.201	18.220	18.201	18.215	18.201
26	20.226	20.221	20.226	20.230	20.226
28	22.167	22.177	22.168	22.180	22.168
30	24.107	24.119	24.107	24.223	24.107
32	26.106	26.091	26.106	26.060	26.106
34	27.911	27.952	27.911	27.936	27.911
36	29.856	29.862	29.856	29.889	29.856
38	31.837	31.847	31.837	31.849	31.837
40	33.726	33.690	33.726	33.890	33.726
mean	14.215	14.215	14.215	14.226	14.213

using the above process. For example, the tree in Figure 3 is the randomly generated tree, then eight sequences can be generated from the tree. Then we use the CCV method [27] to compute the distance between any two DNA sequences, and obtain a distance matrix. The initial sequence S is treated as the outgroup when constructing rooted phylogenetic tree. In total, there are generated 1000 rooted phylogenetic trees for the taxa number n, so 1000 distance matrices are created for each n. In order to better estimate the accuracy of the algorithms, we generate 1000 trees and 1000 distance matrices (both additive and non-additive) for each n. The process is very time-consuming, so we set the taxa number is from 5 to 40 in order to complete them within feasible time.

B. EXPERIMENT

For each generated phylogenetic tree T_1, a distance matrix D will be obtained from T_1. Each algorithm can construct a phylogenetic tree T_2 for D. Smaller the difference between T_1 and T_2 is, more efficient the algorithm is. The difference between two phylogenetic trees is measured by the partition metric of CDRPT software(http://bioinformatics.imu.edu.cn/tree/).

C. RESULTS ON ADDITIVE DISTANCES

Table 2 shows results on the additive distance matrices. For each value n and each algorithm, the value in Table 2 is the average of 1000 distances between the original trees and the phylogenetic trees constructed by the algorithm. The last row in the table shows the average values. Table 2 shows that the constructed phylogenetic trees by NJ, RapidNJ and RandomNJ are the same as the original trees because the distances between them are 0. The distances between the constructed phylogenetic trees by FastJoin and the original trees are smaller than the distances between constructed phylogenetic trees by Clearcut and the original trees in almost all cases. Therefore, it is concluded that NJ, RapidNJ as well as RandomNJ are most efficient among all those algorithms, and FastJoin is more efficient than Clearcut.

D. RESULTS ON NON-ADDITIVE DISTANCES

Table 3 shows the results on the non-additive distance matrices. Similarly, each value in the table is the mean of 1000 distances between the original trees and the phylogenetic trees constructed by the algorithm. The last row in the table shows the average values.

Table 3 shows that the distances between the phylogenetic trees constructed by those algorithms and the original trees have slight difference. From the average values (i.e. values in the last row), it follows that RandomNJ is most efficient in all those algorithms; NJ,FastJoin and RapidNJ with the same performance are more efficient than Clearcut.

V. DISCUSSION

Compared with other algorithms, Clearcut does not have the theoretical evidence on the constructed phylogenetic trees for the additive distance matrices. The experiments have also shown that Clearcut has the worst performance on not only additive distance matrices but also non-additive distance matrices. RandomNJ is a little superior to the others. NJ, RapidNJ and FastJoin have nearly the same performance, which are superior to Clearcut.

REFERENCES

[1] K. Atteson, "The performance of neighbor-joining methods of phylogenetic reconstruction," *Algorithmica*, vol. 25, nos. 2–3, pp. 251–278, Jun. 1999.

[2] S. Besenbacher, L. Mailund, T. Westh-Nielsen, and C. N. S. Pedersen, "RBT—A tool for building refined Buneman trees," *Bioinformatics*, vol. 21, no. 8, pp. 1711–1712, Dec. 2005.

[3] F. Cerutti, L. Bertolotti, T. L. Goldberg, and M. Giacobini, "Taxon ordering in phylogenetic trees by means of evolutionary algorithms," *BioData mining*, vol. 4, no. 1, Jul. 2011, Art. no. 20.

[4] I. Elias and J. Lagergren, "Fast neighbor joining," *Theor. Comput. Sci.*, vol. 410, nos. 21–23, pp. 1993–2000, May 2005.

[5] I. Elias and J. Lagergren, *Fast Neighbor Joining*. Amsterdam, The Netherlands: Elsevier, 2009.

[6] J. Evans, L. Sheneman, and J. Foster, "Relaxed neighbor joining: A fast distance-based phylogenetic tree construction method," *J. Mol. Evol.*, vol. 62, no. 6, pp. 785–792, Jun. 2006.

[7] O. Gascuel, "A note on sattath and tversky's, saitou and nei's, and studier and keppler's algorithms for inferring phylogenies from evolutionary distances," *Mol. Biol. Evol.*, vol. 11, no. 6, pp. 961–963, Nov. 1994.

[8] O. Gascuel, "BIONJ: An improved version of the NJ algorithm based on a simple model of sequence data," *Mol. Biol. Evol.*, vol. 14, no. 7, pp. 685–695, Jul. 1997.

[9] F. Cerutti, L. Bertolotti, T. L. Goldberg, and M. Giacobini, "Taxon ordering in phylogenetic trees: A workbench test," *BMC Bioinf.*, vol. 12, no. 1, Feb. 2011, Art. no. 58.

[10] A. Jain and D. Kihara, "Phylo-PFP: Improved automated protein function prediction using phylogenetic distance of distantly related sequences," *Bioinformatics*, vol. 35, no. 5, pp. 753–759, Mar. 2019.

[11] Q. Jin, I. Grama, C. Kervrann, and Q. S. Liu, "Nonlocal means and optimal weights for noise removal," *SIAM J. Imag. Sci.*, vol. 10, no. 4, pp. 1878–1920, Nov. 2017.

[12] S. Kumar and S. R. Gadagkar, "Efficiency of the neighbor-joining method in reconstructing deep and shallow evolutionary relationships in large phylogenies," *J. Mol. Evol.*, vol. 51, no. 6, pp. 544–553, Dec. 2000.

[13] Y. Liao, M. Lee, C. Ko, and C. Hsiung, "Bioinformatics models for predicting antigenic variants of influenza A/H3N2 virus," *Bioinformatics*, vol. 24, no. 4, pp. 505–512, Feb. 2008.

[14] T. Mailund and C. N. Pedersen, "QuickJoin—Fast neighbour-joining tree reconstruction," *Bioinformatics*, vol. 20, no. 17, pp. 3261–3262, Nov. 2004.

[15] R. Mihaescu, L. Dan, and L. Pachter, "Why neighbor-joining works," *Algorithmica*, vol. 54, no. 1, pp. 1–24, Nov. 2009.

[16] E. K. Molloy and T. Warnow, "Njmerge: A generic technique for scaling phylogeny estimation methods and its application to species trees," in *Comparative Genomics* (Lecture Notes in Computer Science). 2018, pp. 260–276.

[17] E. K. Molloy and T. Warnow, "Treemerge: A new method for improving the scalability of species tree estimation methods," *Bioinformatics*, vol. 35, no. 14, pp. I417–I426, Jul. 2019.

[18] W. R. Pearson, G. Robins, and T. Zhang, "Generalized neighbor-joining: More reliable phylogenetic tree reconstruction," *Mol. Biol. Evol.*, vol. 16, no. 6, pp. 806–816, Jan. 1999.

[19] N. Saitou and M. Nei, "The neighbor-joining method: A new method for reconstructing phylogenetic trees," *Mol. Biol. Evol.*, vol. 4, no. 4, pp. 406–425, Jul. 1987.

[20] L. Sheneman, J. Evans, and J. A. Foster, "Clearcut: A fast implementation of relaxed neighbor joining," *Bioinformatics*, vol. 22, no. 22, pp. 2823–2834, Nov. 2006.

[21] M. Simonsen, T. Mailund, and C. N. S. Pedersen, "Rapid neighbour-joining," in *Proc. Int. Workshop Algorithms Bioinf.*, 2008, pp. 113–122.

[22] M. Simonsen, T. Mailund, and C. N. S. Pedersen, *Inference Large Phylogenies Using Neighbour-Joining*. Berlin, Germany: Springer, 2010.

[23] J. A. Studier and K. J. Keppler, "A note on the neighbor-joining algorithm of saitou and nei," *Mol. Biol. Evol.*, vol. 5, no. 6, pp. 729–730, Jul. 1988.

[24] J. Wang, M. Z. Guo, and L. L. Xing, "Fastjoin, an improved neighbor-joining algorithm," *Genet. Mol. Res.*, vol. 11, no. 3, pp. 1909–1922, Jul. 2012.

[25] J. Wang and M. Z. Guo, "A metric on the space of rooted phylogenetic trees," *Current Bioinf.*, vol. 13, no. 5, pp. 487–491, Nov. 2018.

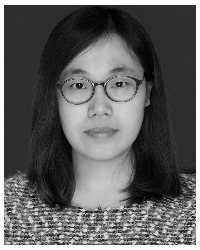
[26] J. Wang and M. Z. Guo, "A review of metrics measuring dissimilarity for rooted phylogenetic networks," *Briefings Bioinf.*, Jul. 2018. doi: 10.1093/bib/bby062.

[27] J. Wang, M. Guo, K. Che, C. Wang, X. Liu, and Y. Liu, "A new distance computing method for DNA sequences in phylogenetic analysis," in *Proc. Int. Conf. Fuzzy Syst. Knowl. Discovery*, Jul. 2013, pp. 690–694.

[28] M. S. Waterman, T. F. Smith, M. Singh, and W. A. Beyer, "Additive evolutionary trees," *J. Theor. Biol.*, vol. 64, no. 2, pp. 199–213, Jan. 1977.

[29] Q. Zou, Q. Hu, M. Guo, and G. Wang, "HAlign: Fast multiple similar DNA/RNA sequence alignment based on the centre star strategy," *Bioinformatics*, vol. 31, no. 15, pp. 2475–2481, Aug. 2015.

[30] Q. Zou, X. B. Li, W.-R. Jiang, Z.-Y. Lin, G.-L. Li, and K. Chen, "Survey of map reduce frame operation in bioinformatics," *Briefings Bioinf.*, vol. 15, no. 4, pp. 637–647, Jul. 2014.



JUAN WANG received the M.S. degree from the Department of Mathematics, Harbin Institute of Technology, in 2009, and the Ph.D. degree from the School of Computer Science and Technology, Harbin Institute of Technology, in 2014. She is currently an Associate Professor with the School of Computer Science, Inner Mongolia University. Her research interests include algorithm design (mainly, construction of phylogenetic trees, and phylogenetic networks) and computational biology (mainly, mathematical models of evolution).