J_2^*: A New Method for Alignment-free Sequence Similarity Measurement

Yue Jiang
College of Mathematics and Informatics
Fujian Normal University
Fuzhou 350108, China.
yueljiang@163.com

Donald A. Adjeroh
Computer Science & Electrical Engineering
West Virginia University
Morgantown, WV 26506, USA.
donald.adjeroh@mail.wvu.edu

Bing-Hua Jiang
Pathology, Anatomy & Cell Biology
Thomas Jefferson University
Philadelphia, PA 19107, USA.
binghjiang@yahoo.com

Jie Lin
College of Mathematics and Informatics
Fujian Normal University
Fuzhou 350108, China.
linjie891@163.com

***Abstract*—Alignment-free sequence comparison methods can compute the similarity of a huge number of sequences much faster than traditional sequence alignment methods. Here, a new non-parametric alignment-free sequence comparison algorithm, J_2^* is proposed to measure sequence similarity based on the suffix tree data structure. Compared against other state-of-the-art alignment-free methods, namely, D_2^z, D_2, D_2^{sh}, D_2^*, WFV, DV, Shi, CPF, DM_k, K_2 and K_2^*, J_2^* has three main advantages: (1) it has the fastest running time in theory and in practice. J_2^* reduces the time for k-words search from $O(N^2)$ to $O(N)$. Our experimental results confirm that it is the fastest among the 11 popular approaches. (2) J_2^* is easy to use: unlike the other alignment-free methods that often need to choose a suitable parameter k, there is no parameter selection for J_2^*. (3) J_2^* does not have any particular requirement for data distribution. Unlike the the parametric methods (such as the D_2-family) that require certain distribution for the data, J_2^* has no demand for a specific input distribution. The improved running time from J_2^* will be very useful in this era of big data, especially, with the increasing data volume of genome sequences.**

***Index Terms*—sequence similarity measurement, alignment-free sequence comparison, suffix tree, biological sequences**

I. Introduction

The evaluation of similarity between sequences is a classical problem that has long been studied in computer science, primarily from the view point of string pattern matching [1], [16]. Given their improved speed, alignment-free methods are gaining more attention than alignment-based methods [42] in evaluating sequence similarity [10], [44]. Alignment-free sequence analysis are widely used in various areas, such as computer science [1], [16], graphics [26], forensic analysis [39], and computational biology. In particular, in genomic sequence analysis, with the increasing size of genome sequences, alignment-free methods are increasingly being used in analyzing DNA sequences [15], [32], [33], RNA sequences [23], [44], and protein sequences [17], [23], [45], as well as in detection of single nucleotide variants in genomes [28], cancer mutations [38], and genetic transfer [8], [10]. Further applications in clinical practice are reported in [7].

Various approaches have been proposed for sequence similarity evaluation (see [1], [16]). Alignment-free sequence comparison methods can be classified into four general types, namely, D_2-statistic family of methods [5], [43], base-base correlations (BBC) [24], feature frequency profiles (FFPs) [11], and compositional vectors (CVs) [49]. The D_2 statistic is reported to be generally faster than the other types [5] since it utilizes the statistical properties of sequence itself.

The proposed J_2^* approach is close in spirit to the D_2 statistic which was first introduced by Blaisdell [4]. Many improved variants have been proposed, such as D_2^z [19], D_2^* [35], and D_2^{sh} [47]. In order to improve its detection power between sequences, D_2^z [19] uses its mean and standard deviation to normalize the D_2 statistic [43]. D_2^* and D_2^{sh} are two other normalization improvement methods which were proposed in [35], [47]. D_2^{sh} is also denoted D_2^S in the literature [35], [43], using an approach based on Shepp statistic [40]. According to a comprehensive review [43], D_2^{sh} and its variant are generally the best D_2 statistical methods for alignment-free comparison of genomic sequences, especially with increasing sequence length.

The D_2-statistic family of algorithms have a general problem of requiring a quadratic or cubic time complexity, with respect to the length of sequences, and the size of k-words being considered. Furthermore, the D_2 family of statistics generally makes some assumptions on the distribution of sequences, for instance, most assumed either a uniform distribution, or a normal distribution, for symbols in sequences.

In this work, a novel nonparametric approach, J_2^* is proposed, which does not make any assumption on its base data distribution. It uses the Jaccard correlation statistic to estimate the similarity between sequences. The Jaccard correlation is

978-1-7281-6215-7/20/$31.00 ©2020 IEEE

a non-parametric method to calculate the correlation between two sets of random variables. It was adopted to measure the similarity among two sets of k-words which are extracted from two sequences respectively. J_2^* reduces the k-words search from $O(N^2)$ to $O(N)$ by using the suffix tree data structure. When compared to the other state-of-the-art alignment-free sequence similarity methods (e.g., D_2, D_2^*, $D_2^{s^h}$, and D_2^z), J_2^* demonstrates an improved power in detecting relatedness between sequences as measured by its ability to generate the correct phylogenetic tree. Further, J_2^* is faster than the other methods, especially since it does not need to search for an optimal length k as the other methods do. This places the proposed J_2^* statistic among the best non-alignment based sequence similarity measures, especially with increasing sequence lengths or increasing size of k-mer.

II. METHODS

A. Basic Notations and Terminology.

Strings. Let $T = T[1 \dots N]$ be the input string of length N, over an alphabet Σ. Let $T = \alpha\beta\gamma$, for some strings α, β, and γ (α and γ could be empty). The string β is called a *substring* of T, α is called a *prefix* of T, while γ is called a *suffix* of T. We will use $T[i]$ to denote the i-th symbol in T.

Suffix tree. Given a string $T = T[1...N]$ with symbols over alphabet Σ, its suffix tree (ST) is a rooted tree with N leaves, where the i-th leaf node corresponds to the i-th suffix ST_i of T $(1 \leq i \leq N)$. Except for the root node and the leaf nodes, each internal node must have at least two descendant children nodes. Each edge in the suffix tree represents a substring of T, and no two edges out of a node start with the same character.

k-word. The k-word (also sometimes called k-mer, or k-gram) is a word (or substring) with length k over an alphabet Σ.

B. Jaccard correlation

The Jaccard correlation is a popular concept in set theory. Given two sets A and B, the union of A and B is $\mathcal{R} = A \bigcup B$. Let $\overline{A}$ denote elements outsides set A. There are three parts in $\mathcal{R}$, namely, $A \bigcap B$, $\overline{A} \bigcap B$, and $A \bigcap \overline{B}$.

In set theory, the Jaccard correlation describes the similarity between two data sets A and B.

$$\begin{aligned} \mathcal{J}(A,B) &= \frac{|A \bigcap B|}{|A \bigcup B|} \\ &= \frac{|A \bigcap B|}{|A \bigcap B| + |\overline{A} \bigcap B| + |A \bigcap \overline{B}|} \\ &= \frac{\min(|A|,|B|)}{\max(|A|,|B|)} \end{aligned} \tag{1}$$

C. Jaccard similarity between two sequences

In order to calculate the Jaccard similarity between two sequences, first, we need to compute the size of the intersection and union for a single k-word w in sequences X and Y. Let X_w and Y_w denote a k-word w in sequences X and Y. Let $|X_w|$ and $|Y_w|$ denote the frequency of a k-word w in sequences X and Y, respectively. For a single k-word w, the Jaccard similarity between sequences X and Y is defined by:

$$\mathcal{J}(X_w, Y_w) = \frac{|X_w \bigcap Y_w|}{|X_w \bigcup Y_w|} = \frac{min(|X_w|, |Y_w|)}{max(|X_w|, |Y_w|)} \tag{2}$$

The above $\mathcal{J}(X_w, Y_w)$ is in the range of $[0, 1]$. When $|X_w| = |Y_w|$, the value is 1, otherwise, if the k-word w does not occur in either sequence (or in both sequences), the value is 0.

Let $\varphi(X_w, Y_w)$ denote the weight of a k-word w over two sequences X and Y. The weight is the ratio of the actual number of occurrences of the k-word w in the concatenated sequence $X\$Y\#$ to the expected frequency in the concatenated sequence $X\$Y\#$. Assume $|\Sigma|$ is the size of the alphabet Σ. For a symbol in the alphabet Σ, the probability of its occurrence at a particular position in the sequence is $\frac{1}{|\Sigma|}$, assuming uniform probability for every symbol. Let $P(w)$ denote the probability of a k-word w. Hence, $P(w) = (\frac{1}{|\Sigma|})^{|w|}$, where $|w|$ denote the length of a k-word w. Let N denote the length of the concatenated sequence $X\$Y\#$, that is, $N = N_X + N_Y + 2$. Thus, the expected number of occurrence $E(w)$ for a k-word w is the length N times the occurring probability $P(w)$. Therefore, the weight $\varphi(X_w, Y_w)$ can be written as in Equation 3:

$$\varphi(X_w, Y_w) = \frac{|X_w| + |Y_w|}{E(w)} = \frac{|X_w| + |Y_w|}{N \times P(w)} = \frac{|X_w| + |Y_w|}{N \times (\frac{1}{|\Sigma|})^{|w|}} \tag{3}$$

where N is the length of the concatenated sequence $X\$Y\#$, that is, $N = N_X + N_Y + 2$; $|\Sigma|$ is the size of the alphabet Σ.

When the length of w is longer, the $E(w)$ will be smaller, the weight $\varphi(X_w, Y_w)$ will be larger. Intuitively, with the increasing length of a k-word, if the k-word occurs in both sequences, the importance of the k-word is increasing. That is, a shorter length k-word occurring in both sequences has less weight.

For a single word w, the weighted Jaccard similarity between sequences X and Y is defined by:

$$\begin{aligned} \mathcal{C}_{\mathcal{J}}(X_w, Y_w) &= \mathcal{J}(X_w, Y_w) \times \varphi(X_w, Y_w) \\ &= \frac{|X_w \bigcap Y_w|}{|X_w \bigcup Y_w|} \times \varphi(X_w, Y_w) \\ &= \frac{min(|X_w|, |Y_w|)}{max(|X_w|, |Y_w|)} \times \frac{|X_w| + |Y_w|}{N \times (\frac{1}{|\Sigma|})^{|w|}} \end{aligned} \tag{4}$$

For a sequence with length N over alphabet Σ, there are $\frac{N(N-1)}{2}$ k-words. Then the Jaccard similarity between two sequences is the sum of the weighted similarity $\mathcal{C}_{\mathcal{J}}(X_w, Y_w)$ for all k-words in two sequences, as shown in Equation 5.

$$\begin{aligned} J_2^*(X,Y) &= \textstyle\sum_{w \in |\Sigma^*|} \mathcal{C}_{\mathcal{J}}(X_w, Y_w) \\ &= \frac{\sum_{k=1}^{\max(N_X, N_Y)} \sum_{w \in \Sigma^k} \mathcal{J}(X_w, Y_w) \times \varphi(X_w, Y_w)}{\sum_{k=1}^{\max(N_X, N_Y)} \sum_{w \in \Sigma^k} \varphi(X_w, Y_w)} \end{aligned} \tag{5}$$

D. Efficient computation of Jacard sequence similarity

Observe that in Equ 5, there are $\sum_{k=1}^{\max(N_X,N_Y)} |\Sigma|^k$ words in the input sequences X and Y. In the concatenated sequence $S = X\$Y\#$ with length $N = N_X + N_Y + 2$, there are at most $\frac{N(N-1)}{2}$ substrings. Hence the number of k-words (substrings with length k) is $O(N^2)$. This implies that the time complexity to compute the Jaccard similarity will be at least $O(N^2)$ by going through all k-words. This complexity is relatively high, especially for very large datasets that are encountered in practice. For the Jacard similarity to be practical, there is a need to find ways to reduce the time complexity. In the following, we show our method to reduce the time complexity to $O(N)$ by using the suffix tree data structure.

Thus, we propose a new alignment-free sequence comparison algorithm, J_2^* to measure sequence similarity. The time complexity for J_2^* is in $O(N)$. We first build a suffix tree from the concatenated sequence $S = X\$Y\#$, and then analyze the frequencies of all k-words in the suffix tree. We describe the main idea below.

First, a suffix tree ST is constructed from the concatenated sequence $S = X\$Y\#$ (length N), there are at most N internal nodes and N leaves, where $N = N_X + N_Y + 2$.

After constructing the suffix tree ST, we need to count the occurrence frequency for each k-word(substring) in concatenated sequence. That is, we need to count the frequency of each k-word in the sequence via the suffix tree. We note that every k-word in the sequence can be identified by starting from the root node in the suffix tree and walking down a path that spells out the substring. There are three cases for different k-words in the tree.

- **Leaf node**: For a k-word ending at a leaf in the suffix tree ST, the occurring frequency is 1.
- **Internal node**: For a k-word ending at an internal node v in the suffix tree ST, the occurring frequency is equal to the total number of leaf nodes in the subtree rooted at node v. For a given node in the suffix tree for sequence T, the path label is also a substring in T. The frequency of this substring(k-word) can be recursively calculated by summing the frequencies along the path of the children nodes.
- **The k-word ending on an edge** $e(u, v)$: for a k-word ending at a character along an edge $e(u, v)$, the occurrence frequency is equal to the frequency of the child node v. All k-words(substrings) ending along the edge from internal node u to v have the same frequency (the same value as the child node v).

We use a bottom-up strategy to calculate substring frequencies in the suffix tree. The time complexity will be $O(N)$, where N is the length of sequence. Here, we have $N = N_X + N_Y + 2$. For a sequence of length N, the total number of internal and leaf nodes is at most $2N$ (usually much less than $2N$). See [1], [16].

E. Algorithm

The proposed J_2^* sequence comparison algorithm is shown in Fig 1 and its corresponding recursive algorithm to calculate the frequency of k-words (substrings) is shown in Fig 2.

In the algorithm (Fig 1), first, a suffix tree data structure ST is constructed from the concatenation string $S = X\$Y\#$ and the corresponding variables are initialized. Then, from Line 2 to Line 4, variables representing each node (v) in the tree are initialized, i.e, $Xcount$, $Ycount$, and $length$. Given a node v, variable $Xcount$ records the number of leaf nodes which come from sequence X, $|X_{w(u,v)}|$, variable $Ycount$ records the number of leaf nodes which come from sequence Y, $|Y_{w(u,v)}|$. Variable $length$ is the length of edge $e(u, v)$, where the node u is the direct parent of node v.

Then, the recursive algorithm $calC_J$ runs to calculate the necessary parameters for each internal node in the tree from Line 5 to Line 7. The algorithm $calC_J$ (Fig 2) processes each internal node in a depth-first search. The depth-first search counts the frequency from bottom-up, which counts the leaf node first (1 for an existing leaf) and then counts the frequency of its direct parent node recursively until reaching the root. The frequency of a direct parent node is the sum of the frequency of all its children nodes. The frequency of edge $e(u, v)$ is the frequency of node v where node v is the direct child node of u.

The sum of C_J and φ parameters are updated accordingly during the processing. After obtaining the sum of C_J and φ, the J_2^* similarity between X and Y is computed by $\frac{\sum C_J}{\sum \varphi}$ in Line 8.

Complexity analysis. The first stage for constructing suffix tree and initializing variables are shown in Line 1-4 in Fig 1. This step uses $O(N)$ space and time. The next stage calls the recursive algorithm $calC_J$ (Fig 2) for each child node in Line 5-7 (Fig 1). In the suffix tree ST, the subtrees never overlap. The total number of subtrees (including leaves) is at most $2N$, where $N = N_X + N_Y + 2$ is the length of the concatenated sequence $S = X\$Y\#$.

The algorithm $calC_J$ (Fig 2) calculates the frequencies for each internal node by using depth-first search. The space and time complexity depend on the number of subtrees. We know that for a sequence of length N, the total number of nodes (both internal nodes and leaf nodes) in the suffix tree is at most $2N$. Thus the space and the time complexity of the second stage is at most $O(2N) = O(N)$.

The last stage just calculates J_2^* in $O(1)$ constant time and space. Taken together, the J_2^* algorithm (Fig 1) runs in $O(N)$ time, and requires $O(N)$ space.

III. Experiments and Analysis

A. Datasets and environment

Two sets of biological sequence data are used on the experiment. The first data set used is the complete mtDNA sequences from [9], [36] containing data on 12 proteins encoded in the H strand of mtDNA in 20 eutherian species, often used to evaluate the similarity of different species, especially using phylogenetic trees. This data set is called the "mtDNA20" dataset.

The second dataset is 23 whole mitochondrial DNA genomes from different Eukaryotic fish species of the suborder

Algorithm: J_2^* algorithm

J2star(X, Y)
1 ST $\leftarrow$ BuildSuffixTree$(X\$Y\#)$; $C_J \leftarrow 0$; $\varphi \leftarrow 0$
2 for each node nd in ST
3 $nd.Xcount \leftarrow 0$; $nd.Ycount \leftarrow 0$;
4 end for
5 for each child node nd in $ST.root$
6 $calC_J(nd, ST.root, C_J, \varphi)$
7 end for
8 $J_2^* \leftarrow \frac{C_J}{\varphi}$
9 return J_2^*

Fig. 1. Algorithm for computing J_2^* similarity.

Algorithm: $CalC_J$ algorithm

calCj$(node, Pnode, C_J, \varphi)$
1 if $node$ is leaf node then do
2 if $node \in X$
3 $Pnode.Xcount++$
4 else
5 $Pnode.Ycount++$
6 end if
7 else
8 for each child node $childnode$ in $node$
9 $calC_J(childnode, node, C_J, \varphi)$
10 $Pnode.Xcount \leftarrow Pnode.Xcount + childnode.Xcount$
11 $Pnode.Ycount \leftarrow Pnode.Ycount + childnode.Ycount$
12 $l_v \leftarrow |E(root, childnode)|$, $l_u \leftarrow |E(root, node)|$
13 $\varphi' \leftarrow \frac{|\Sigma|^{l_v+1}-|\Sigma|^{l_u+1}}{|\Sigma|-1}$
14 $C_J \leftarrow C_J + \varphi' \times \frac{\min(childnode.Xcount, childnode.Ycount)}{\max(childnode.Xcount, childnode.Ycount)}$
15 $\varphi \leftarrow \varphi + \varphi'$
16 end for
17 end if

Fig. 2. Algorithm for computing C_J and φ for each node.

Labroidei, taken from [13]. We could not locate the sequences for two of the species, namely, `P. trewavasae` and `T. moorii`. Thus, though the original work in [13] used 25 species, our dataset contained only 23 of the 25 species. The sequence lengths ranged from 16440 to 17040 symbols. We call this dataset the "Fish23" dataset.

The experiments were performed in a PC environment, running Intel i5, 4 cores, with 16GB RAM and 1TB HD. J_2^* was written using the R Language.

For comparison purposes, we also tested several other state-of-the-art alignment-free methods using the same data sets. The algorithm for D_2 was from [43], D_2^{sh} was from [47], and D_2^* was from [35]. They all were implemented by using C language. The method D_2^z was developed in Perl in the original work of [19]. K_2^* from [22], WFV from [2], DMk from [48], CPF from [3], DV from [51] and Shi from [41].

B. Phylogenetic tree evaluation measure

Here, we compare the phylogentic trees generated by using the distance matrix against the known correct (reference) phylogenetic tree for the species in the data set. Methods that generate trees that are more similar in structure to the reference tree are regarded as having better performance.

The Robinson-Foulds (RF) distance is used to compare the similarity/dissimilarity between two trees [37]. The RF distance (also called the symmetric difference metric) is a well-known approach for measuring the similarity between two trees. In general, a smaller value of the RF distance implies more similarity between the trees.

C. Phylogenetic tree experiment

In the phylogenetic tree experiment, the mtDNA20, and Fish23 datasets are used. The trees published by Cao et al. [9] and by Fischer et al. [13] are used as the reference trees. See also [22], [27].

The experimental results generated for mtDNA20 data are quantitative RF distance which are shown in table I. Methods D_2, D_2^*, D_2^{sh}, D_2^z, K_2, DM_k, CPF, and WFV depend on the input parameter k. Each column contains the distances of a given alignment-free method with parameter k varied from 2 to 9. When $k = 9$, D_2^z generates a runtime error. Thus, we could not obtain a result for this case. Four methods, K_2^*, J_2^*, DV and Shi, do not need to choose any parameter. Results of these four methods are shown in the bottom of table I.

The minimum RF distance in this table is 12 which is present in the column for D_2^{sh} with parameter k=7, 8. The distance of K_2^* and J_2^* method is 12. D_2^* and CPF are able to achieve the second best with $k = 7, 8, 9$ respectively. Hence, the performance advantage of K_2^* and J_2^* are from two main view points: they are not only easy to use (without the necessity to choose a parameter), but also they have better performance.

TABLE I
THE ROBINSON-FOULDS DISTANCE BETWEEN THE REFERENCE PHYLOGENETIC TREE AND PHYLOGENETIC TREES GENERATED USING DIFFERENT ALIGNMENT-FREE STATISTICAL METHODS (WITH $k = 2, 3, \ldots, 9$). RESULTS ARE BASED ON THE MTDNA20 DATASET [9]. D_2^z GENERATED AN ERROR AT $k = 9$. FOUR METHODS, K_2^*, J_2^*, DV AND Shi, WITHOUT VARIED k PARAMETER, THEY ARE REPORTED IN THE LAST ROW FOR BREVITY.

k	D_2	D_2^*	D_2^{sh}	D_2^z	K_2	DMk	CPF	WFV
2	22	26	26	36	26	18	24	26
3	24	26	28	34	22	20	22	24
4	22	20	22	26	22	16	18	24
5	22	20	16	26	20	16	16	22
6	24	16	16	24	18	18	16	24
7	18	14	**12**	20	14	16	14	24
8	18	16	**12**	20	**12**	16	14	24
9	16	14	14	—	**12**	18	16	24
	K_2^*	**12**	DV	20	Shi	22	J_2^*	**12**

Table II shows the RF distances between the reference phylogenetic tree and phylogenetic trees generated using different alignment-free statistical methods (with $k = 2, 3, \ldots 9$) on Fish23 data. In this case, the minimal $RF = 6$ of J_2^* is the best among all. K_2^* is able to achieve the second best of $RF = 8$. D_2, D_2^*, D_2^{sh}, and K_2 are able to obtain the second

best with $RF = 8$ in different k (7,8,9). However, J_2^* and K_2^* are still better selection because they do not need to try all the possible k values (from 2 to 9 in this case). Certainly, J_2^* is the recommended method here for its ease of use, and best reported accuracy (RF=6).

TABLE II
THE ROBINSON-FOULDS DISTANCES BETWEEN THE REFERENCE PHYLOGENETIC TREE AND PHYLOGENETIC TREES GENERATED USING DIFFERENT ALIGNMENT-FREE STATISTICAL METHODS (WITH $k = 2, 3, \ldots 9$). RESULTS ARE BASED ON THE FISH23 DATASET [13]. D_2^z GENERATED AN ERROR AT $k = 9$. FOR BREVITY, THE RESULTS FOR K_2^*, DV, SHI, AND J_2^* ARE REPORTED IN THE LAST ROW(FOUR METHODS WITHOUT PARAMETER k SELECTION PROBLEM).

k	D_2	D_2^*	D_2^{sh}	D_2^z	K_2	DMk	CPF	WFV
2	32	34	36	40	36	30	32	36
3	30	30	28	40	26	28	30	30
4	26	26	30	36	24	22	24	26
5	24	20	22	38	20	20	20	26
6	14	10	20	36	12	10	12	32
7	14	**8**	14	34	**8**	12	12	34
8	**8**	**8**	**8**	34	**8**	12	14	34
9	**8**	10	14	—	10	14	16	34
	K_2^*	**8**	DV	32	Shi	34	J_2^*	**6**

D. Practical running time

Table III shows the comparison of running time using the mtDNA20 data set in generating phylogenetic tree.

The J_2^* method requires $1.13s$ to run. With a quick look at the table, it may appear that this $1.13s$ is slower than the following cases: D_2, D_2^*, and D_2^{sh} methods with $k = 2, 3, 4, 5$. However, as the previous results on two datasets show, these methods reach their best performance with larger values of k, with $k \geq 6$. At these large values of k, all these methods require more time than J_2^* (and DV, Shi, and K_2^* too). Thus, if we consider the need to search through parameter k from 2 to 9 to determine the best one, the running times for $D_2, D_2^*, D_2^{sh}, D_2^z$ will significantly exceed those for J_2^*, and of the other methods that do not depend on varying k values, such as DV, Shi, and K_2^*.

Four methods, K_2^*, DV, Shi, and J_2^*, which do not require parameter selection, consume 3.07, 2.33, 1.37, and 1.13 seconds, respectively. They all run faster than the other methods as they do not need to try all parameters from $k = 2...9$. Considering the performance shown in Table I, we can state that K_2^*(RF=12) and J_2^* (RF=12) are much better than DV(RF=20) and Shi(RF=22). Overall, J_2^* is the best because it has the fastest running time (2.7 times quicker than K_2^*), and also has the best effectiveness, at least in phylogentic tree construction. Similar results were observed for the Fish23 dataset.

IV. CONCLUSION

In this study, a new non-parametric alignment-free sequence comparison algorithm, J_2^* is proposed to measure sequence similarity based on the suffix tree data structure. Comparing to the other 12 state-of-the-art alignment free methods, D_2^z, D_2, D_2^{sh}, D_2^*, WFV, DV, Shi, CPF, DM_k, K_2 and K_2^*, J_2^* has four main advantages: (1) fast running time in theory and in practice; (2) method is easy to use; (3) better performance

TABLE III
PRACTICAL RUNNING TIME (IN SECONDS) USING ALIGNMENT-FREE METHODS ON THE MTDNA20 DATASET IN GENERATING PHYLOGENETIC TREE. D_2^z GENERATED AN ERROR AT $k = 9$. RESULTS OF FOUR METHODS, K_2^*, J_2^*,DV AND Shi WITHOUT PARAMETER k SELECTION PROBLEM, ARE REPORTED IN THE LAST ROW.

k	D_2	D_2^*	D_2^{sh}	D_2^z	K_2	DMk	CPF	WFV
2	0.02	0.05	0.05	1.55	0.41	3.64	13.23	0.004
3	0.03	0.05	0.07	1.56	0.45	4.90	14.02	0.008
4	0.08	0.11	0.15	1.61	0.57	5.91	15.63	0.020
5	0.20	0.34	0.5	1.76	1.94	7.06	16.82	0.088
6	0.56	1.29	2	2.35	2.22	9.78	16.58	0.884
7	1.26	4.91	7	5.38	3.17	18.09	16.43	7.768
8	2.40	18.18	25	19.19	3.63	40.13	16.82	38.3
9	4.58	70.28	99	—	4.34	92.10	17.05	347.1
	K_2^*	3.07	DV	2.33	Shi	1.37	J_2^*	**1.13**

in phylogentic tree construction; (4) unlike the parametric D_2-family that requires uniform or normal distribution of data, J_2^* does not require knowledge of the data distribution. Specifically, J_2^* reduces k-words search from $O(N^2)$ to $O(N)$ in theory which is verified in experiments. J_2^* does not need to search for optimal parameter k as the other alignment-free methods, this greatly improves it flexibility. Comparing to the other state-of-the-art alignment free methods, J_2^* demonstrates an overall superior performance in generating phylogenetic trees, in classifying viral genomes, and in clustering DNA and protein data. The practical running time of J_2^* is the fastest among all the methods, which consistent with its $O(N)$ time bound. This theoretical improvement in time complexity will be of potential benefit for sequence comparison involveing a huge number of genomes, or very large genomes. This is particularly important given the recent rapid increase in the volume of available genomic data sequences.

Acknowledgement: This work is supported in part by the Chinese National Natural Science Foundation (Grant No. 61472082) , Natural Science Foundation of Fujian Province of China (Grant No. 2014J01220), and the US National Science Foundation (Award no.1816005 and no.1920920).

REFERENCES

[1] Adjeroh, D. A., Bell, T., and Mukherjee A. . The Burrows-Wheeler Transform: Data Compression, Suffix Arrays, and Pattern Matching. Springer Publishing Company, Incorporated, 2008.

[2] Bao, J. P., and Yuan, R. Y. . A wavelet-based feature vector model for dna clustering. *Genetics and Molecular Research*, 14(4):19163–19172, 2015.

[3] Bao, J., Yuan, R., and Bao, Z. . An improved alignment-free model for dna sequence similarity metric. *BMC Bioinformatics*, 15(1):321, 2014.

[4] Blaisdell, B. E. . A measure of the similarity of sets of sequences not requiring sequence alignment. *Proc Natl Acad Sci USA*, 83(14):5155–5159, 1986.

[5] Bonham-Carter, O., Steele, J., and Bastola, D. . Alignment-free genetic sequence comparisons: a review of recent approaches by word analysis. *Briefings in Bioinformatics*, 15(6):890–905, 2014.

[6] Borozan, I., Watt, S., and Ferretti, V. . Integrating alignment-based and alignment-free sequence similarity measures for biological sequence classification. *Bioinformatics*, 31(9):1396–404, 2015.

[7] Brittnacher, M. J., Heltshe, S. L., Hayden, H. S., Radey, M. C., Weiss, E. J., Damman, C. J., Zisman, T. L., Suskind, D. L., and Miller, S. I. . Gutss: An alignment-free sequence comparison method for use in human intestinal microbiome and fecal microbiota transplantation analysis. *Plos One*, 11(7):e0158897, 2016.

[8] Bromberg, R., Grishin, N. V., and Otwinowski, Z. . Phylogeny reconstruction with alignment-free method that corrects for horizontal gene transfer. *Plos Computational Biology*, 12(6):e1004985, 2016.

[9] Cao, Y., Janke, A., Waddell, P. J., and Westerman, M., et al. . Conflict among individual mitochondrial proteins in resolving the phylogeny of eutherian orders. *Journal of Molecular Evolution*, 47(3):307–322, 1998.

[10] Cong, Y., Chan, Y. B., and Ragan, M. A. . A novel alignment-free method for detection of lateral genetic transfer based on tf-idf:. *Scientific Reports*, 6:30308, 2016.

[11] Dai, Q., Li, L., Liu, X., Yao, Y., Zhao, F., and Zhang, M. . Integrating Overlapping structures and background information of words significantly improves biological sequence comparison. *Plos One*, 6(11):e26779, 2011.

[12] William H. E. Day. . Optimal algorithms for comparing trees with labeled leaves. *Journal of Classification*, 2(1):7–28, 1985.

[13] Christoph F. , Koblmller Stephan, Glly Christian, et al. . Complete mitochondrial dna sequences of the threadfin cichlid (petrochromis trewavasae) and the blunthead cichlid (tropheus moorii) and patterns of mitochondrial genome evolution in cichlid fishes. *Plos One*, 8(6):e67048, 2013.

[14] Golia, B., Moeller, G. N., Jankevicius, G., Schmidt, A., et al. . Alignment-free formula oligonucleotide frequency dissimilarity measure improves prediction of hosts from metagenomically-derived viral sequences. *Nucleic Acids Research*, 45(1):39–53, 2017.

[15] Guillaume, H., Roland, W., and Jens, S. . Bloom filter trie: an alignment-free and reference-free data structure for pan-genome storage:. *Algorithms for Molecular Biology*, 11(1):3, 2016.

[16] Gusfield, D. . *Algorithms on Strings, Trees and Sequences: Computer Science and Computational Biology*. Cambridge University Press, 1997.

[17] He, L., Li, Y., Rong, L. H., and Yau, S. T. . A novel alignment-free vector method to cluster protein sequences. *Journal of Theoretical Biology*, 427:41, 2017.

[18] Jie, Z., Zhong, P., and Zhang, T. . A novel method for alignment-free dna sequence similarity analysis based on the characterization of complex networks. *Evolutionary Bioinformatics Online*, 12:229, 2016.

[19] Kantorovitz, MR., Robinson, GE., and Sinha, S. . A statistical method for alignment-free comparison of regulatory sequences. *Bioinformatics*, 23:I249C255, 2007.

[20] Karlin, S., Ghandour, G., Ost, F., Tavare, S., and Korn, L. J. . New approaches for computer analysis of nucleic acid sequences. Proceedings of the National Academy of ences of the United States of America, 80(18), 5660-5664, 1983.

[21] Li, Y., He, L., He, R. L., and Yau, S. S. . Zika and flaviviruses phylogeny based on the alignment-free natural vector method. *DNA & Cell Biology*, 36(2):109, 2017.

[22] Lin, J., Adjeroh, D. A., Jiang, B. H., and Jiang, Y. . K_2 and k_2^*: Efficient alignment-free sequence similarity measurement based on kendall statistics. *Bioinformatics*, 34(10):1682-1689, 2018.

[23] Lin, J., Wei, J., Adjeroh, D. A., Jiang, B. H., and Jiang, Y. . SSAW: A new sequence similarity analysis method based on the stationary discrete wavelet transform. *BMC Bioinformatics* 19, 165, 2018.

[24] Liu, Z., Meng, J., and Sun, X. . A novel feature-based method for whole genome phylogenetic analysis without alignment: Application to HEV genotyping and subtyping. *Biochemical and Biophysical Research Communications*, 368(2):223 – 230, 2008.

[25] Luczak, B. B., James, B. T., and Girgis, H. Z. . A survey and evaluations of histogram-based statistics in alignment-free sequence comparison. *Briefings in Bioinformatics*, bbx161, 2017.

[26] Madsen, M. H., Boher, P., Hansen, P. E., and Jrgensen, J. F. . Alignment-free characterization of 2d gratings. *Applied Optics*, 55(2):317, 2016.

[27] Otu H. H., and Khalid, S. . A new sequence distance measure for phylogenetic tree construction. *Bioinformatics*, 19(16):2122–2130, 2003.

[28] Pajuste, F. D. , Kaplinski, L. , Mls, M. , Puurand, T. , Lepamets, M. , and Remm, M. . Fastgt: an alignment-free method for calling common snvs directly from raw sequencing reads. *Scientific Reports*, 7(1):2537, 2017.

[29] Paradis, E., Claude, J., and Strimmer, K. . Ape: Analyses of phylogenetics and evolution in r language. *Bioinformatics*, 20(2):289–290, 2004.

[30] Pattengale, N. D., Gottlieb, E. J., and Moret, B. M. . Efficiently computing the robinson-foulds metric. *Journal of Computational Biology A Journal of Computational Molecular Cell Biology*, 14(6):724–735, 2007.

[31] Pham, D. T., Gao, S., and Phan, V. . An accurate and fast alignment-free method for profiling microbial communities. *Journal of Bioinformatics & Computational Biology*, page 1740001, 2017.

[32] Pizzi, C. . Missmax: alignment-free sequence comparison with mismatches through filtering and heuristics. *Algorithms for Molecular Biology*, 11(1):6, 2016.

[33] Pratas, D. , Silva, R. M. , Pinho, A. J. , and Ferreira, P. J. S. G. . An alignment-free method to find and visualise rearrangements between pairs of dna sequences. *Scientific Reports*, 5, 2015.

[34] Zhang, Q. , Jun, S. R. , Leuze, M. , Ussery, D. , and Nookaew, I. . Viral phylogenomics using an alignment-free method: A three-step approach to determine optimal length of k-mer. *Scientific Reports*, 7:40712, 2017.

[35] Reinert, G., Chew, D., Sun, F., and Waterman Ms. . Alignment-free sequence comparison (i): statistics and power. *Journal of Computational Biology*, 16(12):1615–1634, 2009.

[36] Reyes, A., Gissi, C., Pesole, G., Catzeflis, F. M., and Saccone, C. . Where do rodents fit? Evidence from the complete mitochondrial genome of *sciurus vulgaris*. *Mol. Biol. Evol.*, 17:979–983, 2000.

[37] Robinson D. F., and Foulds, L. R. . Comparison of phylogenetic trees. *Mathematical Biosciences*, 53(1):131–147, 1981.

[38] Justine, R. , Hayssam, S. , Raluca, U. , Bonnefoi H., Richard, I. , and Jonas, B. , et al. . Micado - looking for mutations in targeted pacbio cancer data: An alignment-free method. *Frontiers in Genetics*, 7, 2016.

[39] Sandhya, M., and Munaga, V. N., Prasad, K. . k-nearest neighborhood structure (k-nns) based alignment-free method for fingerprint template protection. In *International Conference on Biometrics*, pages 386–393, 2015.

[40] Shepp, L. . Normal functions of normal random variables. *Siam Review*, 6(4):459–460, 1964.

[41] Shi, L., and Huang, H. . *DNA Sequences Analysis Based on Classifications of Nucleotide Bases*, volume 137. Springer Berlin Heidelberg, 1 edition, 2012.

[42] Smith, T. F., and Waterman, M. S. . Identification of common molecular subsequences. *J Mol Biol*, 147(1):195–197, March 1981.

[43] Kai, S., Jie, R., Gesine, R., Minghua, D., Waterman, M. S., and Fengzhu, S. . New developments of alignment-free sequence comparison: measures, statistics and next-generation sequencing. *Briefings in Bioinformatics*, 15(3):343 – 353, 2013.

[44] Thankachan, S. V. , Chockalingam, S. P. , and Liu, Y. et.al. . A greedy alignment-free distance estimator for phylogenetic inference. *Bmc Bioinformatics*, 18(8):238, 2017.

[45] Tripathi, P., and Pandey, P. N. . A novel alignment-free method to classify protein folding types by combining spectral graph clustering with chou's pseudo amino acid composition. *Journal of Theoretical Biology*, 424:49, 2017.

[46] Wagner, R. A., and Fischer, M. J. . The string-to-string correction problem. *Journal of the ACM*, 21(1):168–173, 1974.

[47] Wan, L., Reinert, G., Sun, F., and Waterman, M. S. . Alignment-free sequence comparison (ii): theoretical power of comparison statistics. *Journal of Computational Biology A Journal of Computational Molecular Cell Biology*, 17(11):1467–1490, 2010.

[48] Wei, D., Jiang, Q., Wei, Y., and Wang, S. . A novel hierarchical clustering algorithm for gene sequences. *BMC Bioinformatics*, 13(1):1–15, 2012.

[49] Wu, X., Wan, X., Wu, G., Xu, D., and Lin, G. . Phylogenetic analysis using complete signature information of whole genomes and clustered neighbour-joining method. *International Journal of Bioinformatics Research and Applications*, 2(3):219–248, 2006.

[50] Yaveroglu, O. N., Milenkovic, T., and Przulj, N. . Proper evaluation of alignment-free network comparison methods. *Bioinformatics*, 31(16):2697–704, 2015.

[51] Zhao, B., He, R. L., and Yau, S. T. . A new distribution vector and its application in genome clustering. *Molecular Phylogenetics & Evolution*, 59(2):438–443, 2011.