2012 14th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing

Clustering Methods based on Closest String via Rank Distance

Liviu P. Dinu and Radu-Tudor Ionescu
Department of Computer Science
University of Bucharest
14 Academiei, Bucharest, Romania
E-mails: ldinu@fmi.unibuc.ro, raducu.ionescu@gmail.com

***Abstract*—This paper aims to present two clustering methods based on rank distance. Rank distance has applications in many different fields such as computational linguistics, biology and informatics. Rank distance can be computed fast and benefits from some features of the edit (Levenshtein) distance.**

In [1] two clustering methods based on rank distance are described. The K-means algorithm uses the median string to represent the centroid of a cluster, while the hierarchical clustering method joins pairs of strings and replaces each pair with the median string. Two similar clustering algorithms are about to be presented in this paper, only that the closest string will be considered instead of the median string.

The new clustering algorithms are compared with those presented in [1] and other similar clustering techniques. Experiments using mitochondrial DNA sequences extracted from several mammals are performed to compare the results of the clustering methods. Resultsmdemonstrate the clustering performance and the utility of the new algorithms.

***Keywords*-clustering; k-means; hierarchical clustering; rank distance; closest string; closest substring; DNA; DNA applications; DNA sequencing; bioinformatics; phylogeny.**

I. Introduction

Clustering has an important role in a wide variety of fields: biology, statistics, pattern recognition, information retrieval, machine learning, data mining, psychology and other social sciences. There are various clustering algorithms that differ significantly in their notion of what constitutes a cluster. The appropriate clustering algorithm and parameter settings (such as the distance function to use, the density threshold or the number of expected clusters) depend on the individual data set. But not all clustering algorithms can be applied on a particular data set. For example, clustering methods that depend on a standard distance function (such as K-means) cannot be applied on a data set of objects (such as strings) for which a standard distance function cannot be computed.

The goal of this work is to introduce two clustering algorithms for strings similar to the algorithms presented in [1]. Note that a few distance measures for strings can be considered, but we focus on using a single distance that has very good results in terms of accuracy and time for many important problems. The distance we focus on is termed *rank distance* [2] and it has applications in biology [3**?**], natural language processing [4], authorship attribution [5]. The first clustering algorithm presented here is a centroid model that represents each cluster by a single closest string. The second one is a connectivity model the builds a hierarchy of clusters based on distance connectivity. Both use rank distance.

Let us describe the paper organization. In section II we introduce notation and mathematical preliminaries. We first introduce the rank distance and then we define closest string problem. Section III gives an overview of related work regarding sequencing and comparing DNA, rank distance and clustering methods. The clustering algorithms are described in section IV. The experiments using mitochondrial DNA sequences from mammals are presented in section V. Finally, we draw our conclusion and talk about further work in section VI.

II. Preliminaries

A ranking is an ordered list and is the result of applying an ordering criterion to a set of objects. Formally,

Definition 1: Let $\mathcal{U} = \{1, 2, ..., \#\mathcal{U}\}$ be a finite set of objects, named universe (we write $\#\mathcal{U}$ for the cardinality of $\mathcal{U}$). A *ranking* over $\mathcal{U}$ is an ordered list: $\tau = (x_1 > x_2 > ... > x_d)$, where $x_i \in \mathcal{U}$ for all $1 \leq i \leq d$, $x_i \neq x_j$ for all $1 \leq i \neq j \leq d$, and $>$ is a strict ordering relation on the set $\{x_1, x_2, ..., x_d\}$.

A ranking defines a partial function on $\mathcal{U}$ where for each object $i \in \mathcal{U}$, $\tau(i)$ represents the position of the object i in the ranking τ. Observe that the objects with high rank in τ have the lowest positions.

The rankings that contain all the objects of an universe $\mathcal{U}$ are termed *full rankings*, while the others are *partial rankings*. We define the order of an object $x \in \mathcal{U}$ in a ranking σ of length d, by $ord(\sigma, x) = |d+1-\sigma(x)|$. By convention, if $x \in \mathcal{U} \setminus \sigma$, we have $ord(\sigma, x) = 0$.

Definition 2: Given two partial rankings σ and τ over the same universe $\mathcal{U}$, we define the rank distance between them as:

$$\Delta(\sigma, \tau) = \sum_{x \in \sigma \cup \tau} |ord(\sigma, x) - ord(\tau, x)|.$$

The author of [2] proves that Δ is a distance function. Rank distance is an extension of the Spearman footrule distance [6], defined below.

Definition 3: If σ and τ are two permutations of the same length, then $\Delta(\sigma, \tau)$ is named the Spearman footrule distance.

Rank distance is naturally extended to strings. The following observation is immediate: if a string does not contain identical symbols, it can be transformed directly into a ranking (the rank of each symbol is its position in the string). Conversely, each ranking can be viewed as a string, over an alphabet equal to the universe of the objects in the ranking. The next definition formalizes the transformation of strings that have identical symbols into rankings.

Definition 4: Let n be an integer and let $w = a_1 \dots a_n$ be a finite word of length n over an alphabet Σ. We define the extension to rankings of w, $\bar{w} = a_{1,i(1)} \dots a_{n,i(n)}$, where $i(j) = |a_1 \dots a_j|_{a_j}$ for all $j = 1, \dots n$ (i.e. the number of occurrences of a_j in the string $a_1 a_2 \dots a_j$).

Example 1: If $w = aaababbbac$ then $\bar{w} = a_1a_2a_3b_1a_4b_2b_3b_4a_5c_1$.

Observe that given $\bar{w}$ we can obtain w by simply deleting all the indexes. Note that the transformation of a string into a ranking can be done in linear time (by memorizing for each symbol, in an array, how many times it appears in the string [**?**]). We extend the rank distance to arbitrary strings as follows:

Definition 5: Given $w_1, w_2 \in \Sigma^*$, we define $\Delta(w_1, w_2) = \Delta(\bar{w}_1, \bar{w}_2)$.

Example 2: Consider the following two strings $x = abcaa$ and $y = baacc$. Then, $\bar{x} = a_1b_1c_1a_2a_3$ and $\bar{y} = b_1a_1a_2c_1c_2$. Thus, the rank distance between x and y is the sum of the absolute differences between the orders of the characters in $\bar{x}$ and $\bar{y}$:

$$\Delta(x, y) = |1-2| + |4-3| + |5-0| + |2-1| + |3-4| + |0-5| = 14$$

The computation of the RD between two rankings can be done in linear time in the cardinality of the universe. Our universe has precisely $|w_1| + |w_2|$ objects and, thus, the rank distance between w_1 and w_2 can be computed in linear time.

Let χ_n be the space of all strings of size n over an alphabet Σ and let $p_1,\ p_2, \dots, p_k$ be k strings from χ_n. The closest string problem (CSP) is to find the center of the sphere of minimum radius that includes all the k strings. An alternative formulation of the problem is to find a string from χ_n which minimizes the distance to all the input strings.

Problem 1 (Closest string via rank distance): Let $P = \{p_1, p_2, \dots, p_k\}$ be a set of k length n strings over an alphabet Σ. The *closest string problem via rank distance (CSRD)* is to find a minimal integer d (and a corresponding string t of length n) such that the maximum rank distance from t to any string in P is at most d. We say that t is the closest string to P and we name d the radius. Formally, the goal is to compute:

$$\min_{x \in \chi_n} \max_{i=1..k} \Delta(x, p_i).$$

III. Related Work

A. Sequencing and Comparing DNA

Biologists have spent many years creating a taxonomy (hierarchical classification) of all living things: kingdom, phylum, class, order, family, genus, and species. Thus, it is perhaps not surprising that much of the early work in cluster analysis sought to create a discipline of mathematical taxonomy that could automatically find such classification structures. More recently, biologists have applied clustering to analyze the large amounts of genetic information that are now available.

In many important problems in computational biology a common task is to compare a new DNA sequence with sequences that are already well studied and annotated. Sequences that are similar would probably have the same function, or, if two sequences from different organisms are similar, there may be a common ancestor sequence [7]. Another important problem with practical motivations for biologists is related to the finding of motifs or common patterns in a set of given DNA sequences. A typical case where the last mentioned problem occurs is, for example, when one needs to design genetic drugs with structure similar to a set of existing sequences of RNA [8].

The standard method used in computational biology for sequence comparison is by sequence alignment. Algorithmically, the standard pairwise alignment method is based on dynamic programming [9]. Although dynamic programming for sequence alignment is mathematically optimal, it is far too slow for comparing a large number of bases, and too slow to be performed in a reasonable time. Also, since some of the search solutions are inaccurate from a biological point of view, alternative approaches periodically are explored in computational biology. This important problem, known also as DNA sequence comparison, is ranked in the top of two lists with major open problems in bioinformatics [10, 11].

The standard distances with respect to the alignment principle are edit (Levenshtein) distance [12] or its ad-hoc variants. The study of genome rearrangement [13] was investigated also under Kendall tau distance (the minimum number of swaps needed to transform a permutation into the other).

Traditionally, the Closest String Problem (CSP) is related to Hamming distance and it all started from a code theory application [14]. There are recent studies that investigate CSP under Hamming distance with advanced programming techniques such as integer linear programming (ILP) [15].

When CSP emerged in bioinformatics, the problem was investigated from many points of view. These investigations implied the use of different distances. The most intensive studied approach was the one based on edit distance. The study of genome rearrangement specific problems lead to the development of new problems related to closest string via various distances used in the investigations of this problems. Recently, in [16] it is shown that the CSP via swap distance (or Kendall distance) and CSP via element duplication distance (the element duplication distance between w_1 and w_2 is the minimum number of element duplications needed to transform a string w_2 into a string w_1) remain NP-hard too.

B. Rank Distance

To measure the similarity between strings Dinu proposes a new distance measure, termed *rank distance (RD)* [2], with applications in biology [3, 17], natural language processing [4], authorship attribution [5]. Rank distance can be computed fast and benefits from some features of the edit distance.

To measure the distance between two strings with RD we scan (from left to right) both strings and for each letter from the first string we count the number of elements between its position in the first string and the position of its first occurrence in the second string. Finally, we sum up all these scores and obtain the rank distance. In other words, the rank distance measures the "gap" between the positions of a letter in the two given strings, and then sums up these values. Intuitively, the rank distance gives us the total non-alignment score between two sequences.

Note that in the Hamming and rank distance case the median string problem is tractable [18], while in the edit distance case it is NP-hard. In [19] it is shown that the CSRD is NP-hard. In the clustering algorithms that are about to be presented in this paper, CSRD is important because steps in both algorithms involve finding the closest string for a cluster (subset) of strings. To find the closest string we use the genetic algorithm described in [20].

Theoretically, RD works on strings from any alphabet (finite or infinite). But, in many practical situations the alphabet is of fixed constant size (in computational biology, the DNA and protein alphabets are respectively of size 4 and 20). For some applications, one needs to encode the DNA or protein sequences on a binary alphabet that expresses only a binary property of the molecule, e.g. hydrophoby (for instance, this is the case in some protocols that identify similar DNA sequences [21]). In [22, 23] it is shown that closest string and median string are NP-hard for finite and even binary alphabets. The existence of fast exact algorithms, when the number of input strings is fixed, is investigated in [22].

C. Clustering Methods

In recent years considerable effort has been made for improving algorithm performance of the existing clustering algorithms. The authors of [24] propose an extensions to the K-means algorithm for clustering large data sets with categorical values. Another unsupervised data mining algorithm (called BIRCH) used to perform hierarchical clustering over particularly large data-sets is presented in [25].

With the recent need to process larger and larger data sets (also known as big data), the willingness to treat semantic meaning of the generated clusters for performance has been increasing. This led to the development of pre-clustering methods such as canopy clustering [26], which can process huge data sets efficiently.

A standard method of hierarchical clustering that uses rank distance is presented in [17]. It presents a phylogenetic tree of several mammals comparable to the structure of other trees reported by other researches [27, 28]. Two clustering algorithms for strings that are based on rank distance are described in [1]. Note that the next section introduces two similar algorithms, only that the closest string will be considered instead of the median string.

IV. Clustering Methods based on Closest String via Rank Distance

Cluster analysis (or clustering) groups data objects based only on information found in the data that describes the objects and their relationships. The goal is that the objects within a group (or cluster) are similar (or related) to one another and different from the objects in other groups. The greater the similarity within a group and the greater the difference between groups, the better or more distinct the clustering. The clusters should capture the natural structure of the data.

There are various clustering algorithms that differ significantly in their notion of what constitutes a cluster. Popular notions of clusters include groups with low distances among the cluster members, dense areas of the data space, intervals or particular statistical distributions. In this section we introduce two clustering algorithms that are based on a distance measure for strings. These algorithms are to be applied on data sets that contain objects represented as strings, such as text, DNA sequences, etc.

A. K-Means-Type Algorithm based on Rank Distance

The K-means clustering technique is a simple method of cluster analysis which aims to partition a set of objects into K clusters in which each object belongs to the cluster with the nearest mean. The algorithm begins with choosing K initial centroids, where K is an *a priori* parameter, namely, the number of clusters desired. Each object is then assigned to the nearest centroid, and each group of objects assigned to a centroid is a cluster. The centroid of each cluster is then updated based on the objects assigned to that cluster. We repeat the assignment and update steps until no point changes clusters or until a maximum number of iterations is reached.

If string objects are considered for clustering, then we need a way to determine the centroid string for a certain cluster of strings. We propose the use of the closest (or centre) string computed with rank distance. In the implementation presented [1] the median string is used instead of the closest string. Note that computing the closest (centre) string for a cluster of strings is equivalent to solving Problem 1. We also use rank distance to assign strings to the nearest centre string. Thus, the algorithm that we propose is entirely based on rank distance.

Algorithm 1 partitions a set of strings into K clusters using rank distance. It aims at minimizing an objective function, in this case given by

$$J = \sum_{j=1}^{k} \max_{i=1}^{n} \Delta(x_i^{(j)}, c_j),$$

where $\Delta(x_i^{(j)}, c_j)$ is the rank distance between a string $x_i^{(j)}$ in cluster j and the cluster centroid c_j. The objective function is simply an indicator of the distance of the input strings from their respective cluster centers.

Algorithm 1: K-means type algorithm based on rank distance

```
1. Initialization: Randomly select $K$ strings as initial centroids.
2. Loop: Repeat until centroids do not change or until a maximum number of iterations is reached
    a. Form $K$ clusters by assigning each string to its nearest centre string.
    b. Recompute the centroid (centre string) of each cluster using rank distance.
```

Note that the algorithm proposed here is only similar to a K-means clustering approach since the choice of clusters centroids is fundamentally different from a standard K-means. Thus, we refer to Algorithm 1 as a K-means-type algorithm. Although Algorithm 1 will always terminate, it does not necessarily find the most optimal configuration, corresponding to the global objective function minimum. The algorithm is also significantly sensitive to the initial randomly selected centroids. However, it can be run multiple times to reduce this effect.

Regarding computational complexity, the K-means algorithm proposed here depends on the complexity of the algorithm that computes the closest string. We used the genetic algorithm described in [3] that computes the median string in $O(n^3)$ time, where n is the string length. If we denote the maximum number of iterations by m, then the complexity of Algorithm 1 is $O(m \times K \times n^3)$. Note that the genetic algorithm gives only a near optimal solution for the closest string. This is the best solution available since closest string problem is NP-hard.

B. Hierarchical Clustering based on Rank Distance

Many hierarchical clustering techniques are variations on a single algorithm: starting with individual objects as clusters, successively join the two nearest clusters until only one cluster remains. These techniques connect objects to form clusters based on their distance. Note that these algorithms do not provide a single partitioning of the data set, but instead provide an extensive hierarchy of clusters that merge with each other at certain distances. This structure can be represented using a dendrogram.

Apart from the choice of a distance function, another decision is needed for the linkage criterion to be used. Because a cluster consists of multiple objects, there are multiple candidates to compute the distance to. Popular choices are single-linkage, complete-linkage, or average-linkage.

A standard method of hierarchical clustering that uses rank distance is presented in [17]. It presents a phylogenetic tree of several mammals comparable to the structure of other trees reported by other researches [27, 28]. Our belief that a hierarchical clustering method designed to deeply integrate rank distance would perform better was empirically demonstrated in [1]. Here we propose a similar hierarchical clustering method that also works only for strings. It uses rank distance, but instead of a linkage criterion we use something different, that is to determine a centroid string for each cluster and join clusters based on the rank distance between their centroid strings. In the implementation of Algorithm 2 we choose the closest string as cluster centroid. To compute the closest string is equivalent to solving Problem 1. In the implementation presented [1] the median string is used instead of the closest string.

Algorithm 2: Hierarchical clustering based on rank distance

```
1. Initialization: Compute rank distances between all initial strings.
2. Loop: Repeat until only one cluster remains
    a. Join the nearest two clusters to form a new cluster.
    b. Determine the closest string of the new cluster.
    c. Compute rank distances from this new closest string to existing closest strings (of previously formed clusters).
```

The analysis of the computational complexity of Algorithm 2 is straightforward. If we consider m to be the number of the initial strings, then $O(m^2)$ time is required to compute rank distances between them. The algorithm builds a binary tree structure where the leaves are the initial m strings. Thus, it creates $m - 1$ intermediate clusters until only one cluster remains. The most heavy computational step is to determine the closest string of a cluster which takes $O(n^3)$ time, where n is the string length. Usually n is much greater than m and the algorithm complexity in this case is $O(m \times n^3)$.

V. Experiments

A. Data set

Our clustering methods are tested on a classical problem in bioinformatics: the phylogenetic analysis of the mammals. In the experiments presented in this paper we use mitochondrial DNA sequence genome of the following 22 mammals available in the EMBL database: human (*Homo sapiens*, V00662), common chimpanzee (*Pan troglodytes*, D38116), pigmy chimpanzee (*Pan paniscus*, D38113), gorilla (*Gorilla gorilla*, D38114), orangutan (*Pongo pygmaeus*, D38115), sumatran orangutan (*Pongo pygmaeus abelii*, X97707), gibbon (*Hylobates lar*, X99256), horse (*Equus caballus*, X79547), donkey (*Equus asinus*, X97337), Indian rhinoceros (*Rhinoceros unicornis*, X97336), white rhinoceros (*Ceratotherium simum*, Y07726), harbor seal (*Phoca vitulina*, X63726), gray seal (*Halichoerus grypus*, X72004), cat (*Felis* *catus*, U20753), fin whale (*Balenoptera physalus*, X61145), blue whale (*Balenoptera musculus*, X72204), cow (*Bos taurus*, V00654), sheep (*Ovis aries*, AF010406), rat (*Rattus norvegicus*, X14848), mouse (*Mus musculus*, V00711), North American opossum (*Didelphis virginiana*, Z29573), and platypus (*Ornithorhyncus anatinus*, X83427). Mitochondrial DNA (mtDNA) is the DNA located in organelles called mitochondria. The DNA sequence of mtDNA has been determined from a large number of organisms and individuals, and the comparison of those DNA sequences represents a mainstay of phylogenetics, in that it allows biologists to elucidate the evolutionary relationships among species. In mammals, each double-stranded circular mtDNA molecule consists of $15,000$-$17,000$ base pairs.

In our experiments each mammal is represented by a single mtDNA sequence that comes from a single individual. We mention that DNA from two individuals of the same species differs by only $0,1\%$. This means, for example, that mtDNA from two different humans differs by less than 20 base pairs. Because this difference cannot affect our study, we conduct the experiments using a single mtDNA sequence for each mammal.

B. K-means Experiment

The first experiment is to cluster the 22 mammalian DNA sequences using K-means clustering into 6 clusters. The clustering is relevant if the 6 clusters match the 7 families of mammals available in our data set: Primates, Perissodactylae, Cetartiodactylae, Rodentia, Carnivora, Metatheria, Monotremata. Note that we have only one member of the Metatheria family, that is the North American opossum, and only one member of the Monotremata family, that is the platypus. The opossum and the platypus should be clustered together or with members of the Rodentia family.

As in the experiment performed in [1], we use only the first 1000 nucleotides extracted from each of DNA sequence so that we can compare the two K-means methods. The clustering results of the two methods are shown on columns in Table I.

The K-means algorithm based on closest string is able to clearly distinguish the Perissodactylae family. It only leaves the sheep out of the Cetartiodactylae family. The Primates and Carnivora families are joined together in a single cluster. The human is surprisingly placed in single cluster completely separated from all other mammals. If we join clusters with labels 2 and 4 we obtain the Rodetina family clustered together with the opossum and the platypus. In consequence, Algorithm 1 is able to create relevant groups for most of the families, even if we only use the first 1000 nucleotides.

On the other hand, the K-means algorithm based on the median string is able to clearly distinguish the Carnivora and Primates families. It also leaves the sheep out of the Cetartiodactylae family. The Rodentia and Metatheria families are joined together in a single cluster, while the Perissodactylae family is separated in two clusters.

Table I
K-means based on median string (K-median) vs. K-means based on closest string (K-closest): clustering results for 22 DNA sequences using rank distance.

Mammal	Family	K-median	K-closest
Cow	Cetartiodactylae	2	1
Sheep	Cetartiodactylae	0	2
Big whale	Cetartiodactylae	2	1
Fin whale	Cetartiodactylae	2	1
Cat	Carnivora	4	0
Gray seal	Carnivora	4	0
Harbour seal	Carnivora	4	0
Opossum	Metatheria	3	4
Platypus	Monotremata	0	2
Human	Primates	5	3
Gibbon	Primates	5	0
Gorilla	Primates	5	0
Pygmy chimpanzee	Primates	5	0
Orangutan	Primates	5	0
Chimpanzee	Primates	5	0
Sumatran orangutan	Primates	5	0
Horse	Perissodactylae	0	5
Donkey	Perissodactylae	0	5
Indian rhinoceros	Perissodactylae	1	5
White rhinoceros	Perissodactylae	1	5
House mouse	Rodentia	3	4
Rat	Rodentia	3	2

Overall, we believe the algorithm introduced in this paper gives similar results to the algorithm presented in [1]. The algorithm based on the median string seems to perform slightly better, but the two techniques should be compared on more datasets for a strong conclusion.

C. Hierarchical Clustering Experiment

The second experiment is to apply the hierarchical clustering proposed in section IV on the 22 mammalian DNA sequences. The resulted phylogenetic tree is going to be compared with phylogenetic trees obtained with standard hierarchical clustering method, such as the ones presented in [1, 17].

The phylogenetic tree obtained with Algorithm 2 is presented in Figure 1. It was obtained using only the first 3000 nucleotides of each DNA sequence.

Analyzing the phylogenetic tree in Figure 1 we observe that our hierarchical clustering method is a able to separate the Primates, Perissodactylae, Carnivora and Cetartiodactylae families. The house mouse is joined together with the opossum and the platypus in a cluster that is relevant from a biological point of view, since they are related mammals. The only confusion of our clustering method is that the rat is clustered with the sheep instead of the house mouse. To our surprise, we obtained the exact tree structure as the hierarchical clustering from [1]. This means that the two hierarchical clustering methods are very similar as they are able to capture the same structure of the data.

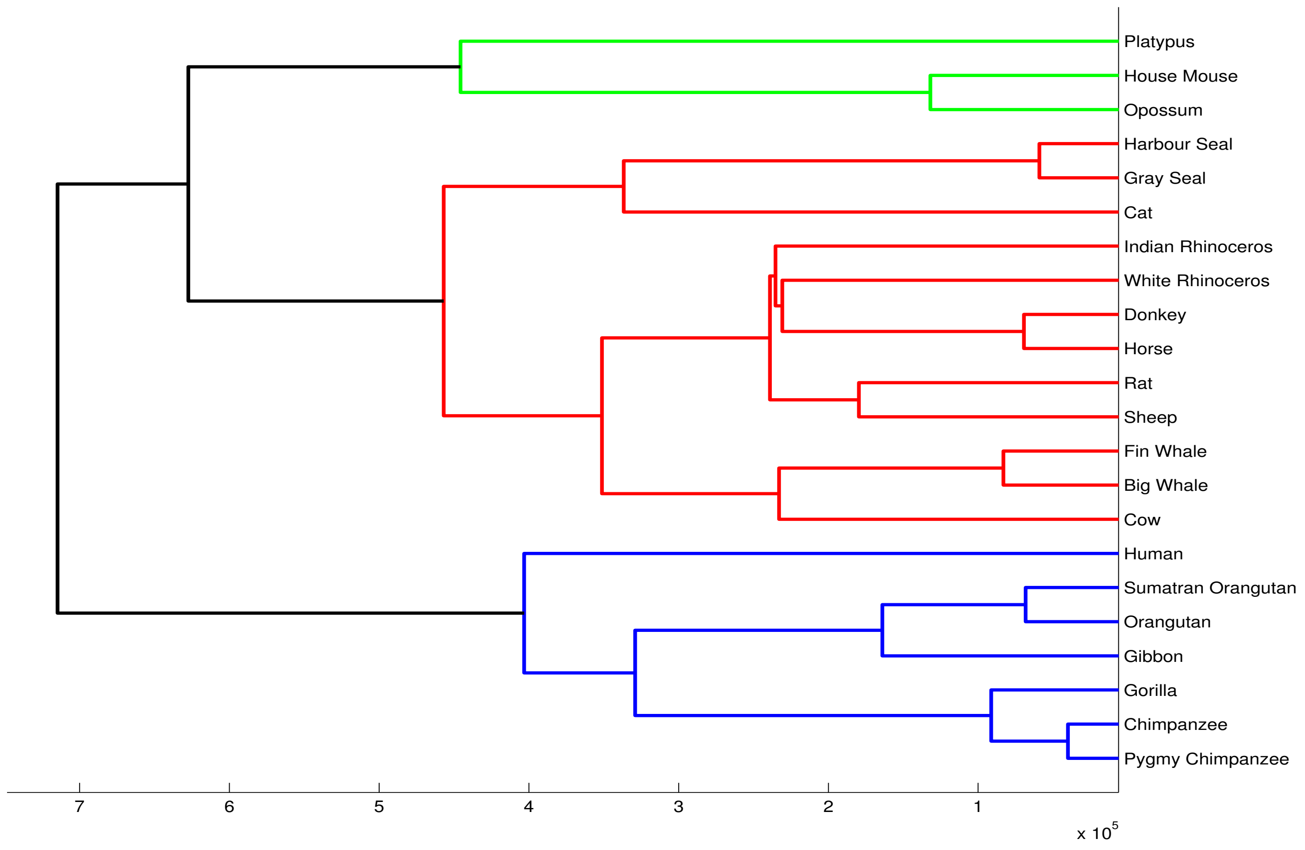

Figure 1. Phylogenetic tree obtained from mammalian mtDNA sequences using rank distance.

By contrast to our result, in the phylogenetic tree presented in [17] the Perissodactylae and Carnivora families are mixed together. The standard hierarchical clustering method used in [17] is also unable to join the rat with the house mouse. Therefore, we state that the hierarchical clustering method presented in this paper performs better, despite that it uses only the first 3000 nucleotides instead of the entire mtDNA sequences.

If we look at a higher level in the phylogenetic tree obtained with Algorithm 2, the Primates are distinct from all the other mammals. But, what is really interesting is that the human is very different from the other Primates. If we choose a cutting point that would separate the dendrogram in 7 clusters, then the human falls in a separate cluster. In other words, there will be a cluster that contain a single representative DNA sequence, that of the human's. This is consistent with the result of the K-means-type algorithm that also placed the human in a separate cluster.

VI. Conclusion and Further Work

In this paper we exibit two clustering methods based on rank distance. The first one is a K-means-type algorithm that represents each cluster by a single closest string. The second one is a connectivity model the builds a hierarchy of clusters based on distance between the closest strings of clusters. Both compute the closest string of a cluster using rank distance which has very good properties when compared to other distance measures.

Our experiments on the phylogenies of mammals produced results which are similar or sometimes better to those reported in the literature [1, 17, 27]. The experiments also demonstrate the utility of the two algorithms proposed in this paper.

Note that the hierarchical clustering method presented here uses rank distance, but instead of a linkage criterion we proposed to determine the closest string for each cluster and join clusters based on the rank distance between their closest strings. In the future we want to try another implementation that uses both the closest string and the median string via a consensus rule. We also plan to perform more experiments with our new clustering methods for strings.

Acknowledgment

The contribution of the authors to this paper is equal. The authors thank to the anonymous reviewers for helpful comments.

References

[1] L. P. Dinu and R. T. Ionescu, "Clustering based on Rank Distance with Applications on DNA," *In Proceedings of ICONIP*, vol. 7663, 2012.

[2] L. P. Dinu, "On the classification and aggregation of hierarchies with different constitutive elements," *Fundamenta Informaticae*, vol. 55, no. 1, pp. 39–50, 2003.

[3] L. P. Dinu and R. T. Ionescu, "An efficient rank based approach for closest string and closest substring," *PLoS ONE*, vol. 7, no. 6, p. e37576, 06 2012.

[4] A. Dinu and L. P. Dinu, "On the syllabic similarities of romance languages," *In Proceedings of CICLing*, vol. 3406, pp. 785–788, 2005.

[5] L. P. Dinu, M. Popescu, and A. Dinu, "Authorship Identification of Romanian Texts with Controversial Paternity," *In Proceedings of LREC*, 2008.

[6] P. Diaconis and R. L. Graham, "Spearman footrule as a measure of disarray," *Journal of Royal Statistical Society. Series B (Methodological)*, vol. 39, no. 2, pp. 262–268, 1977.

[7] A. W. Liew, H. Yan, and M. Yang, "Pattern recognition techniques for the emerging field of bioinformatics: A review," *Pattern Recognition*, vol. 38, no. 11, pp. 2055–2073, Nov. 2005.

[8] K. J. Lanctot, M. Li, B. Ma, S. Wang, and L. Zhang, "Distinguishing string selection problems," *Information and Computation*, vol. 185, no. 1, pp. 41–55, Aug. 2003.

[9] T. Smith and M. Waterman, "Comparison of biosequences," *Advances in Applied Mathematics*, vol. 2, no. 4, pp. 482–489, Dec. 1981.

[10] E. V. Koonin, "The emerging paradigm and open problems in comparative genomics," *Bioinformatics*, vol. 15, pp. 265–266, 1999.

[11] J. C. Wooley, "Trends in computational biology: a summary based on a recomb plenary lecture," *Journal of Computational Biology*, vol. 6, pp. 459–474, 1999.

[12] V. I. Levenshtein, "Binary codes capable of correcting deletions, insertions and reverseals," *Cybernetics and Control Theory*, vol. 10, no. 8, pp. 707–710, 1966.

[13] J. Palmer and L. Herbon, "Plant mitochondrial DNA evolves rapidly in structure, but slowly in sequence," *Journal of Molecular Evolution*, vol. 28, pp. 87–89, 1988.

[14] M. Frances and A. Litman, "On covering problems of codes," *Theory Comput. Syst.*, vol. 30, no. 2, pp. 113–119, 1997.

[15] M. Chimani, M. Woste, and S. Bocker, "A closer look at the closest string and closest substring problem," *In Proceedings of ALENEX*, pp. 13–24, 2011.

[16] Y. V. Popov, "Multiple genome rearrangement by swaps and by element duplications," *Theoretical Computer Science*, vol. 385, no. 1-3, pp. 115–126, 2007.

[17] L. P. Dinu and A. Sgarro, "A Low-complexity Distance for DNA Strings," *Fundamenta Informaticae*, vol. 73, no. 3, pp. 361–372, 2006.

[18] L. P. Dinu and F. Manea, "An efficient approach for the rank aggregation problem," *Theoretical Computer Science*, vol. 359, no. 1-3, pp. 455–461, 2006.

[19] L. P. Dinu and A. Popa, "On the closest string via rank distance," *In Proceedings of CPM*, vol. 7354, pp. 413–426, 2012.

[20] L. P. Dinu and R. Ionescu, "A genetic approximation for closest string via rank distance," *Proceedings of SYNASC*, pp. 207–215, 2011.

[21] D. J. States and P. Agarwal, "Compact encoding strategies for dna sequence similarity search," *In Proceedings of the 4th International Conference on Intelligent Systems for Molecular Biology*, pp. 211–217, 1996.

[22] F. Nicolas and E. Rivals, "Complexities of centre and median string," vol. 2676, pp. 315–327, 2003.

[23] ——, "Hardness Results for the Center and Median String Problems under the Weighted and Unweighted Edit Distances," *Journal of Discrete Algorithms*, vol. 3, pp. 390–415, 2005.

[24] Z. Huang, "Extensions to the k-means algorithm for clustering large data sets with categorical values," *Data Mining Knowledge Discovery*, vol. 2, no. 3, pp. 283–304, 1998.

[25] T. Z. Tian, R. Ramakrishnan, and M. Livny, "Birch: an efficient data clustering method for very large databases," *SIGMOD Rec.*, vol. 25, no. 2, pp. 103–114, 1996.

[26] A. McCallum, K. Nigam, and L. H. Ungar, "Efficient clustering of high-dimensional data sets with application to reference matching," *In Proceedings of ACM SIGKDD*, pp. 169–178, 2000.

[27] A. Reyes, C. Gissi, G. Pesole, F. M. Catzeflis, and C. Saccone, "Where Do Rodents Fit? Evidence from the Complete Mitochondrial Genome of Sciurus vulgaris," *Mol. Biol. Evol.*, vol. 17, no. 6, pp. 979–983, 2000.

[28] Y. Cao, A. Janke, P. J. Waddell, M. Westerman, O. Takenaka, S. Murata, N. Okada, S. Paabo, and M. Hasegawa, "Conflict among individual mitochondrial proteins in resolving the phylogeny of Eutherian orders," *J. Mol. Evol.*, vol. 47, pp. 307–322, 1998.