A New Iterative Algorithm for Computing a Quality Approximate Median of Strings based on Edit Operations.

J. Abreu[*,a], J. R. Rico-Juan[b]

[a]Dpto Informática, Universidad de Matanzas, Carretera a Varadero Km. 3 1/2, Matanzas, Cuba
[b]Dpto Lenguajes y Sistemas Informáticos, Universidad de Alicante, San Vicente del Raspeig, Alicante, Spain

Abstract

This paper presents a new algorithm which can be used to compute an approximation to the median of a set of strings. The approximate median is obtained through the successive improvements of a partial solution. The edit distance from the partial solution to all the strings in the set is computed in each iteration, thus accounting for the frequency of each of the edit operations in all the positions of the approximate median. A goodness index for edit operations is later computed by multiplying their frequency by the cost. Each operation is tested, starting from that with the highest index, in order to verify whether applying it to the partial solution leads to an improvement. If successful, a new iteration begins from the new approximate median. The algorithm finishes after all the operations have been examined without a better solution being found. Comparative experiments involving Freeman chain codes encoding 2D shapes and the Copenhagen chromosome database show that the quality of the approximate median string is similar to benchmark approaches but achieves a much faster convergence.

Key words: approximate median string, edit distance, edit operations

1. Introduction

Extending the concept of “median” to structural representations such as strings has been a challenging issue in Pattern Recognition for some time, as is shown in the review presented in Jiang et al. (2004). This problem arises in many applications such

[*]Corresponding author. Fax:965909326
Email addresses: jose.abreu@umcc.cu (J. Abreu), juanra@dlsi.ua.es (J. R. Rico-Juan)

as 2D shape representation and prototype construction (Jiang et al., 2000; Bunke et al., 2002), the clustering of strings (Lourenço and Fred, 2005), Self-Organized Maps of strings (Kohonen, 1998; Fischer and Zell, 2000) or the combination of multiple source translations (González-Rubio and Casacuberta, 2010).

Formally, given a set $S = \{S_1, S_2, ..., S_n\}$ of strings over the alphabet $\sum$ and a distance function $D(S_i, S_j)$ which measures the dissimilarity between strings S_i and S_j, the distance from a string S' to all strings in S can be computed by the expression (1).

$$SOD(S') = \sum_{S_i \in S} D(S', S_i) \quad (1)$$

The *median string* is the string $\hat{S} \in \sum^*$ that minimizes (1). This string is also denoted as the *generalized median string*. A common approximation to the true median string is the *set median*, a string in S which minimizes (1). It is not necessary for either the median string or the set median to be unique.

An exact algorithm with which to compute the median of a set of strings was proposed by Kruskal (1983). However, in most practical applications this is not a suitable approach due to the high computational time requirements. As Casacuberta and Antonio (1997) and Nicolas and Rivals (2005) point out, there are various formulations of this problem within the NP-Complete class. Several approximations have therefore been proposed. One approach that has been studied by several authors is that of building the approximate median by using the successive changes of an initial string. Perturbations can be applied one or more at a time, as in the works of Martínez-Hinarejos et al. (2003) and Fischer and Zell (2000), respectively. The results of empirical testing show that the first approach leads to high quality approximations but requires more computational time. The principal motivation of this work is to describe a new algorithm that is able to compute a quality approximation to the median string like that of Martínez-Hinarejos et al. (2003), but requires significantly less computational effort. In the Section 2 some related works are examined. Section 3 describes the proposed approach and provides an analysis of the computational cost bounds for the algorithm. Various comparative experiments are described in Section 4, and finally Section 5 shows our conclusions and some lines for further research.

2. Related works

Many approximated solutions have been described since Kruskal (1983) proposed an exact algorithm that could be used to compute the median string for a given set S of N strings of a length of l and the Levenshtein (Levenshtein, 1966) metric. This algorithm runs in $O(l^N)$ proportional time. A number of heuristics therefore address this difficulty by reducing the size of the search space. Some authors, such as Olivares and Oncina (2008), have studied the approximation to the median string not only under the Levenshtein edit distance but also under the stochastic edit distance (Ristad and Yianilos, 1998). In other works, the search for the approximate median is not performed directly in the string space but rather in a vectorial space in which the strings are embedded; this is the approach studied in Jiang et al. (2012) which also relies on the weighted median concept described by Bunke et al. (2002).

One general strategy is to construct the approximate median letter by letter from an initial empty string. In order to decide which symbol is the next to be appended it is necessary to define a goodness function. The greedy procedure described in Casacuberta and Antonio (1997) implements this approach. An improvement to the aforementioned method is described in Kruzslicz (1999) through the use of a refined criterion which allows the next letter to be selected. Another approach that has been studied by several authors is that of building the approximate median by using successive perturbations of an initial string. Two important issues regarding this kind of method are how to select a perturbation leading to an improvement and how to make the algorithm converge faster without spoiling the results. Another interesting topic is that of studying the effect of performing modifications one by one or simultaneously. Kohonen (1985) starts from the set median and systematically changes the guess string by applying insertions, deletions and substitutions in every position. In Martínez-Hinarejos et al. (2003) the authors proposes to improve a partial solution $\hat{S}$ generating new candidates by applying all possible substitutions, insertions or deleting the symbol at a position i. The new partial solution is the string, selected from all the new candidates and $\hat{S}$, which minimizes (1). This procedure is repeated for every position i. The effect of choosing a different initial string as the set median or a greedy approximation is also studied. Theoretical and empirical results show that this method is capable of achieving very good approximations to the true median string. Note that these methods do not define a criterion to compare the operations in order to select which one can lead to better results in each case. In Martínez-Hinarejos et al. (2002) authors describe alternatives to speed up the computation of the approximated median string. Based on information provided by the weight matrix used to compute the edit distance certain operations are preferred instead others. For example, no all possible substitutions are tried but only the two closest symbols to the one in the analysed position.

Some heuristic knowledge which can help to assess how promising a modification will be are included in Fischer and Zell (2000) and Mollineda (2004). The quality of a partial solution $\hat{S}$ is evaluated by computing its distance from every string in the set, and it is thus also possible to discover the sequences of edit operations. In an attempt to speed up the convergence of the search procedure, these authors propose the simultaneous performance of several modifications by applying the most frequent edit operation, including "do nothing" in each position of the partial solution. This process is repeated while modifications increase the quality of the partial solution.

This approach has two potential drawbacks. Applying the most common operation in every position does not guarantee the best results, and a further issue is that although it might be relatively simple to figure out how applying just one operation will affect $SOD(\hat{S})$, this does not hold when several changes are made at the same time. For example, let $\hat{S}$ be a partial solution and op_i be an edit operation which occurs several times when computing the distance from a partial solution to strings in S. Op_i thus determines a subset $S^{YES} \subseteq S$ of those strings in which op_i occurs when computing the distance from $\hat{S}$. There is also another set $S^{NO} = S - S^{YES}$. Let $\hat{S}'$ be a new solution after applying op_i to $\hat{S}$. Intuitively, it may be expected that the distance from $\hat{S}'$ to strings in S^{YES} decreases as regards $\hat{S}$. A formal discussion of this result can be found in Bunke et al. (2002). The effect on the strings in S^{NO} clearly needs to be taken into account. Since sets induced by each operation may be different when applying multiple operations, it might be very difficult to characterize the effect on $SOD(\hat{S})$. Empirical results, which will be discussed later, suggest that methods which apply multiple perturbations at the same time are able to find a better approximation than the set median very quickly. However, approaches which perform modifications one by one, such as Martínez-Hinarejos et al. (2003), significantly outperform the former methods with respect to the average distance to the set of the approximate median computed.

3. A new algorithm for computing a quality approximate median string

As was noted earlier, a general scheme that can be used to search for an approximate median string is:

- select an initial coarse approximation to the median, as the set median.
- generate a new solution by performing some modifications to the current solution.
- repeat while a particular modification leads to an improvement or another stop condition holds.

The works commented on Section 2 suggest that when it is necessary to find a quality approximation to the median string, applying modifications one by one would appear to be a better strategy. The theoretical results in Jiang and Bunke (2002) and Martínez-Hinarejos (2003) show that the approximation computed by the algorithm proposed in Martínez-Hinarejos et al. (2003) is very close to the lower bound obtained for the value of $SOD(\hat{S})$ for the true median.

3.1. Computing the approximate median string

The algorithm in Martínez-Hinarejos et al. (2003) tests every possible operation in each position of the partial solution, and it might therefore be very useful to study how to reduce the size of the search space without spoiling the quality of results, which is one of the principal motivations of this work. The proposed algorithm is based on two main ideas:

- selecting the appropriate modification by paying attention to certain statistics from the computation of the edit distance from the partial solution to every string in the set.

- applying modifications one by one.

Heuristic information could help to avoid to test a number of useless solutions, which would reduce the amount of times that $SOD(\hat{S})$ is evaluated. Another distinctive feature is that if the best operation according to the goodness index does not lead to an improvement, other low ranked operations can be tested.

The *AppMedianString* procedure outlines how to compute the approximate median string.

3.2. Selecting the best edit operation

In our case, the most suitable edit operation in step t will be selected by examining two approaches. The first simply implies ranking operations by their *frequency* while computing the edit distance from the partial solution to strings in the set, as in Fischer and Zell (2000). Note that the selected operation is that with the best overall ranking,

Function AppMedianString(S,R) :$\hat{S}$

```
/* S:  instance set to compute the approximate median.                         */
/* R:  initialization string.                                                   */
R' = R;
repeat
    Ŝ = R';
    foreach instance s_i ∈ S do
        compute D(R', s_i);
        obtain that Q_si^R' is the minimum cost edit sequence needed to transform
            R' into s_i;
        update statistics for the operation in each position j of R';
    end foreach
    let O_p be an operation queue sorted by its goodness index;
    /* Generate new candidates R' while none of them improve Ŝ                   */
    while ∑_{s_i∈S} D(Ŝ, s_i) ≤ ∑_{s_i∈S} D(R', s_i) and O_p ≠ ∅ do
        op_i = O_p.dequeue;
        obtain a new candidate R' applying op_i to Ŝ;
    end while
until no operation op_i applied to Ŝ improve the result;
return Ŝ;
```

not the most frequent in a specific position. However, under a more general weighting scheme for edit operations, the frequency might not be the best assessment of how promising a transformation is. We therefore propose the use of *Frequency* × *Cost* as a goodness index. For example, let $\hat{S}^t$ be the candidate solution and $S = \{S_1, S_2, S_3\}$. Without loss of generality, let us suppose that the best ranked edit operation (op_1) is a substitution with a frequency of 2, and cost of 1. Let us also suppose that there is another substitution (op_2) with a frequency of 1 but with a cost of 3. From the results in Bunke et al. (2002) we obtain that an $\hat{S}^{t+1}$ built by applying op_1 will satisfy $D(\hat{S}^{t+1}, S_1) = D(\hat{S}^t, S_1) - 1$ and $D(\hat{S}^{t+1}, S_2) = D(\hat{S}^t, S_2) - 1$. Regardless of the value of $D(\hat{S}^{t+1}, S_3)$ it can be expected that $SOD(\hat{S})$ will decrease by 2. A similar analysis shows that the application of op_2 leads to a reduction of 3.

3.3. An illustrative example

The following example illustrates the algorithm's behavior. Let $\hat{S}^t = \{5, 5, 0\}$, $S_1 = \{3, 1, 1, 2\}$ and $S_2 = \{0, 6, 1, 6\}$. The substitution of a symbol a for b obtain the cost $min\{|a-b|, 8-|a-b|\}$, while insertions and deletions obtain the cost of 2. Table 1 shows the computation of the edit distance from $\hat{S}^t$ to S_1 and S_2. In the first case, this results in one of the optimal edit sequences $\{s(5, 3), s(5, 1), s(0, 1), i(2)\}$. $D(\hat{S}^t, S_2)$ results in $\{s(5, 0), s(5, 6), s(0, 1), i(6)\}$. Table 2 shows an edit operation ranked by its frequency. Note how a different goodness index leads to a different ranking. Applying the best operation $s(0, 1)$ in position 3 results in $\hat{S}^{t+1} = \{5, 5, 1\}$, which improves $SOD(\hat{S})$ since $D(\hat{S}^{t+1}, S_1) = 8$ and $D(\hat{S}^{t+1}, S_2) = 6$. If the best operation does not lead to an improvement, then the second best option must be tested, and so on. Note that in the list of perturbations there may be different operations related to the same position. This option does not occur in Fischer and Zell (2000) and Mollineda (2004). The process is repeated by starting from the new solution while some operations lead to a better approximation. The example above also shows how ranking by *Frequency* $\times$ *Cost* can lead to better results. As was explained previously, by applying $s(0, 1)$ we obtain $SOD(\hat{S}^{t+1}) = 14$. The last column in the Table 2 shows that the operations may be ranked differently. In this case, $s(5, 1)$ in position 2 is the operation with the best goodness index. If it were to be applied, then $\hat{S}^{t+1} = \{5, 1, 0\}$ and thus $D(\hat{S}^{t+1}, S_1) = 5$ and $D(\hat{S}^{t+1}, S_2) = 5$, which is $SOD(\hat{S}^{t+1}) = 10$.

3.4. Computational cost analysis

The procedure used to compute the approximate median string needs to compute the distance from the partial solution to every string in the set. Under the Levenshtein edit distance this can be carried out in time $O(l^2)$ by using the dynamic programming algorithm presented in Wagner and Fischer (1974), where l is the length of the longest string. The **foreach** statement loops N times, and the first stage of the algorithm thus

Table 1: Computation of the edit distance cost from $\hat{S}^t = \{5,5,0\}$ to $S_1 = \{3,1,1,2\}$ and $S_2 = \{0,6,1,6\}$. Substitutions of a symbol a by a symbol b have cost $min\{|a-b|, 8-|a-b|\}$ while deletions and insertions have cost of 2. An optimal path is shaded in order to follow the best cost operations easily and visually.

(a)

		3	1	1	2
	0	2	4	6	8
5	2	2	4	6	8
5	4	4	6	8	9
0	6	6	5	7	9

(b)

		0	6	1	6
	0	2	4	6	8
5	2	3	3	5	7
5	4	5	4	6	6
0	6	4	6	5	7

Table 2: Ranking of edit operations

Operation	Position	Frequency	Frequency × Cost
s(0,1)	3	2	2
s(5,0)	1	1	3
s(5,1)	2	1	4
s(5,6)	2	1	1
s(5,3)	1	1	2
i(2)	3	1	2
i(6)	3	1	2

requires a time that is proportional to $O(N \times l^2)$. Assuming that no perturbations improve the solution, the inner **while** loop needs to examine the whole queue O_p.

Let $|\sum|$ be the size of the alphabet; $min\{N, |\sum|\}$ substitutions are possible for each of the l symbols in $\hat{S}$, this is the maximum number of substitutions, and there are thus $O(l \times min\{N, |\sum|\})$ potential substitutions. The same result holds for insertions. Only l deletions are possible. A pessimistic upper bound to $|O_p|$ is therefore $O(2 \times l \times min\{N, |\sum|\} + l)$. In the worst case, each operation in O_p involves computing the distance from R' to all the strings, which requires $O(N \times l^2)$. Under these assumptions, inner **while** takes a time proportional to $O(N \times l^3 \times min\{N, |\sum|\})$. Let k be the number of times that the outer **repeat** loops, thus the algorithm requires $O(k \times N \times l^3 \times min\{N, |\sum|\})$, which is the same time required by the algorithm described by Martínez-Hinarejos et al. (2001). However, in practice the proposed approach behaves much better as suggest results which will be discussed in Section 4.

4. Experimental results

Experiments were carried out to evaluate the performance of the proposed approach when computing an approximate median string. To ensure independent results with regard to the alphabet, the strings over two sets of symbols were tested. In the first case, $\sum = \{0, 1, 2, 3, 4, 5, 6, 7, \lambda\}$, corresponding to the directions of Freeman chain codes (Freeman, 1974) where λ denotes the empty symbol used for deletions and insertions. Edit operation costs were fixed in a manner similar to that of Rico-Juan and Micó (2003), that is, a cost of 2 for deletions and insertions and $min\{|a - b|, 8 - |a - b|\}$ for substitutions. The strings in each set are not randomly generated but are a chain code representation of the contours from two widely known 2D shape databases, the *NIST-3 Uppercase Letters* and the *USPS Digits* , (Jain and Zongker, 1997; García-Díez et al., 2011; Rico-Juan and Iñesta, 2012), with 26 and 10 classes, respectively. Four independent samples of 20 instances per class were drawn for a total of 144 different sets. Our approach was used to compute an approximate median for each of them. The proposed algorithm, referred to as *JR-S*, was compared to the methods proposed by Fischer and Zell (2000) and Mollineda (2004) which performs several modifications at the same time, and that of Martínez-Hinarejos (2003) which modifies the partial solution in a one by one manner.

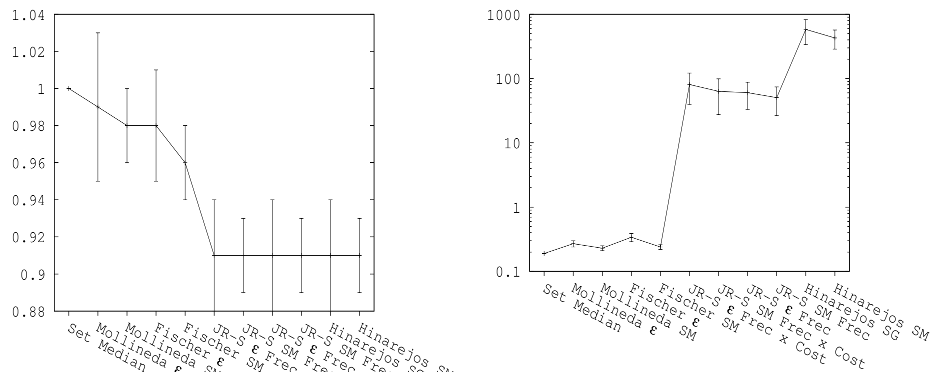
In a second test, strings were drawn from the chromosomes dataset used by Martínez-Hinarejos et al. (2003). This time $\sum = \{a, b, c, d, e, =, A, B, C, D, E, \lambda\}$, and the cost of each operation was computed as in Martínez-Hinarejos et al. (2003). Four samples of 20 instances were again selected for each of the 22 classes.

Tables 3 and 4 show the results for each set in the respective databases. In order to facilitate the comparison of the results of different algorithms and datasets, in each case we compute the ratio $\frac{SOD(\hat{S})}{SOD(S^M)}$, where S^M is the set median. The lower it is, the better the approximation to the true median found by the algorithm is. In each case "ε", "S^M" or "S^G" refer to the initial string, that is, the empty string, the set median and the greedy initialization proposed by Casacuberta and Antonio (1997). Since all the algorithms in the test work in an iterative manner, the number of distances computed by each approach that evaluates $SOD(\hat{S})$ was also studied. The graphics in Figure 1 and 2 show the average value for $\frac{SOD(\hat{S})}{SOD(S^M)}$ and the average number of distances computed by each approach in all the experiments.

Besides, a third experiment was carried out to compare the results with respect to the true median. In this case we collect four sets of 20 random generated strings over the alphabet $\sum = \{0, 1, 2, 3, 4, 5, 6, 7, \lambda\}$ with length varying from 3 to 8. Operation costs were fixed as explained before. Table 5 shows results on this simple database.

As mentioned previously, the results confirm that applying perturbations to the partial solution one by one leads to a much better quality approximation to the true median in terms of $SOD(\hat{S})$. In every set, either the proposed approach or Martínez-Hinarejos (2003) provides the most precise approximation. In general, the solutions computed with *JR-S* are equivalent to or even better than those attained with Martínez-Hinarejos (2003) but, as Tables 3 and 4 show, the proposed approach is, on average, about 10 times faster than Martínez-Hinarejos (2003) in terms of the computed distances. In some cases ranking the operations by *Frequency* × *Cost* instead *Frequency* can lead to slightly better approximations, but in general, it also requires the computation of additional distances. On the other hand, although its results are not so good in terms of the approximate median quality in the methods of Fischer and Zell (2000) and Mollineda (2004), only a few distances are needed to notably improve the set median. In both cases it would appear that the algorithm gets stuck in a local minimum after a small number of iterations.

A comparison in terms of running time was also included, as Figure 3 shows. The experiments were performed in a computer with an Intel X5355-2.66 GHz CPU (4 cores) and 8 Gb RAM. It can be observed that algorithms introduced by Fischer and Zell (2000) and Mollineda (2004) are in average about 30 times faster than ours. On the other hand, the proposed approach runs near 8 times faster than the methods described by Martínez-Hinarejos et al. (2003).

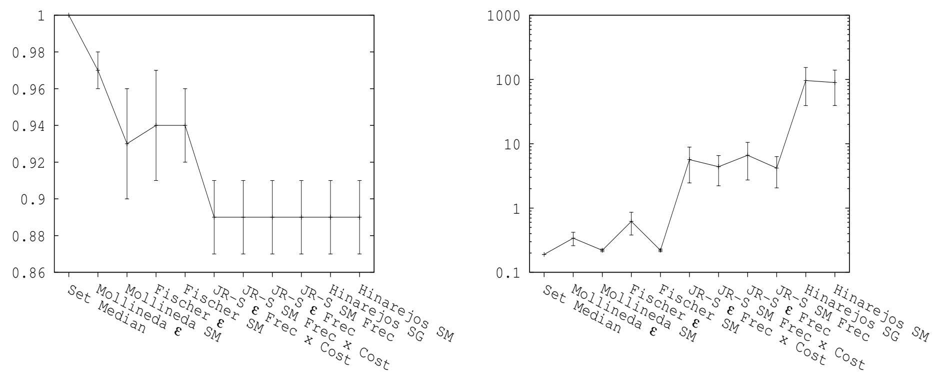


(a) (b)

Figure 1: 1a shows the average for $\frac{SOD(\hat{S})}{SOD(S^M)}$ in all experiments. This measure represents the quality of the results. The chart in 1b shows the average number of distances (in thousands) (Freeman chain codes set). In both cases, less value is better.

5. Conclusions and Future work

A new approach with which to compute a quality approximation to the median string has been presented. The algorithm builds an approximate median through the successive refinements of a partial solution. Modifications are applied one by one in a manner similar to that of Martínez-Hinarejos et al. (2003), and empirical results show that this approach leads to better approximations than those methods which apply several perturbations simultaneously, although the latter runs much faster. Comparisons with Martínez-Hinarejos (2003) show that the proposed algorithm is able to compute high-quality approximations to the true median string, but requires significantly less computation, and is about 10 times faster, which makes it highly suitable for applications that require a precise approximation. As was pointed in Section 2, an operation op_i determines two subsets S^{YES} and S^{NO} from S. Applying op_i to $\hat{S}$ results in new string $\hat{S}'$ such as the distance from strings in S^{YES} to $\hat{S}'$ will decrease. Further research



(a) (b)

Figure 2: 2a shows the average for $\frac{SOD(\hat{S})}{SOD(S^M)}$ in all experiments. This measure represents the quality of the results. The chart in 2b shows the average number of distances (in thousands) (Copenhagen chromosomes set). In both cases, less value is better.

may address to better characterize how behaves the distance from $\hat{S}'$ to strings in S^{NO} without computing those distances, but using information gathered when computing the distances to $\hat{S}$. This can help to select the better operation to reduce the number of distances computed without spoiling the approximation quality. Another subject of interest is that of analysing how the choice of a different optimal path will affect results, since a different ranking might be obtained.

Acknowledgements

This work is partially supported by the Spanish CICYT under project DPI2006-15542-C04-01, the Spanish MICINN through project TIN2009-14205-CO4-01 and by the Spanish research program Consolider Ingenio 2010: MIPRCV (CSD2007-00018).

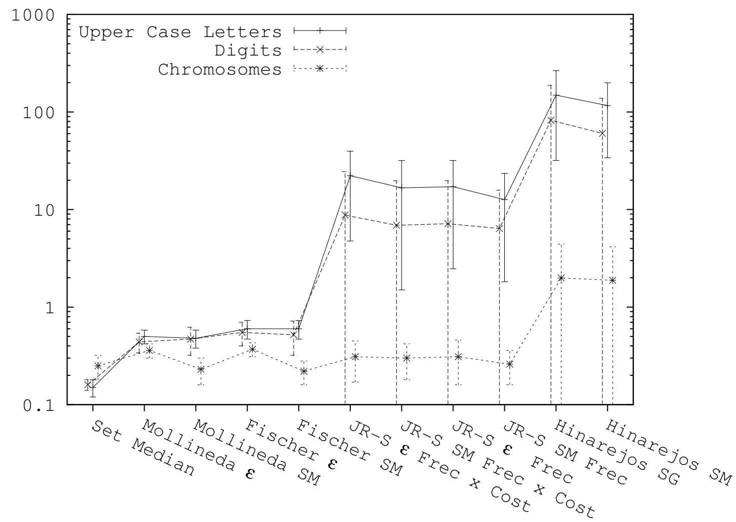


Figure 3: Average running time, in seconds, for each algorithm in each database experiments.

Table 3: Average distance from the approximated median to each string in the set. (Freeman chaincodes)

Class	Set Median	Mollineda ε	Mollineda S^M	Fischer ε	Fischer S^M	JR-S ε Freq × Cost	JR-S S^M Freq × Cost	JR-S ε Freq	JR-S S^M Freq	Hinarejos S^G	Hinarejos S^M
A	102.0±5.0	98.7±6.0	98.0±6.0	99.3±5.0	96.4±5.0	92.2±5.0	92.4±5.0	92.1±5.0	92.2±5.0	92.3±5.0	92.1±5.0
B	120.5±4.0	122.3±7.0	119.7±4.0	120.8±5.0	119.0±4.0	110.3±5.0	110.1±5.0	110.2±5.0	110.2±5.0	110.2±5.0	110.1±5.0
C	116.2±13.0	110.7±10.0	111.6±12.0	109.9±11.0	109.9±11.0	103.6±10.0	103.5±10.0	103.3±10.0	103.3±10.0	103.3±10.0	103.3±10.0
D	126.3±29.0	131.5±29.0	126.3±29.0	128.8±30.0	126.3±29.0	117.1±28.0	117.2±28.0	117.1±28.0	117.0±28.0	117.0±28.0	117.2±28.0
E	181.8±12.0	191.1±19.0	176.0±8.0	174.7±7.0	172.9±7.0	168.5±12.0	165.9±8.0	168.3±12.0	165.5±8.0	168.6±12.0	165.5±8.0
F	154.2±8.0	149.0±10.0	148.1±8.0	146.2±9.0	145.9±7.0	137.0±7.0	137.1±7.0	136.9±7.0	137.0±7.0	137.0±7.0	137.2±7.0
G	190.1±17.0	184.7±17.0	184.8±16.0	180.9±17.0	180.5±16.0	171.0±16.0	170.2±16.0	170.8±16.0	170.6±16.0	170.6±16.0	170.4±16.0
H	193.3±19.0	192.5±20.0	193.1±19.0	189.2±19.0	189.0±19.0	176.7±18.0	176.5±18.0	176.6±18.0	176.6±19.0	176.6±19.0	176.6±18.0
I	141.6±17.0	155.1±20.0	141.6±17.0	150.0±19.0	141.6±17.0	137.3±18.0	137.3±18.0	137.7±18.0	137.6±18.0	137.5±18.0	137.3±18.0
J	184.9±14.0	180.1±13.0	182.0±17.0	182.6±8.0	178.6±13.0	167.0±12.0	167.2±12.0	166.9±11.0	166.9±11.0	167.0±11.0	166.9±11.0
K	197.1±17.0	195.4±21.0	190.7±17.0	186.8±17.0	186.6±15.0	175.1±14.0	175.0±15.0	175.1±15.0	175.5±14.0	179.3±20.0	175.2±15.0
L	89.7±5.0	89.4±4.0	86.9±6.0	87.8±5.0	85.9±6.0	82.4±5.0	82.7±5.0	82.4±5.0	82.4±5.0	82.5±5.0	82.5±5.0
M	225.6±16.0	220.2±11.0	218.4±11.0	214.6±14.0	214.1±12.0	201.4±11.0	201.4±11.0	201.2±11.0	201.5±11.0	201.6±12.0	201.4±11.0
N	191.8±11.0	192.1±10.0	190.7±10.0	188.8±10.0	187.9±11.0	178.8±9.0	178.8±9.0	178.9±9.0	178.6±9.0	179.0±9.0	178.7±9.0
O	75.4±11.0	78.5±16.0	74.8±12.0	76.8±14.0	74.6±12.0	70.7±12.0	70.7±12.0	70.8±12.0	70.6±12.0	70.6±12.0	70.4±12.0
P	100.5±10.0	100.9±13.0	100.4±10.0	99.1±13.0	99.5±11.0	92.2±10.0	92.1±10.0	92.0±10.0	92.2±10.0	92.3±10.0	92.3±10.0
Q	109.7±9.0	115.0±10.0	109.4±8.0	107.0±8.0	107.3±8.0	100.4±7.0	100.5±7.0	100.2±7.0	100.3±7.0	100.2±7.0	100.2±7.0
R	150.5±10.0	149.0±11.0	150.0±10.0	145.9±12.0	147.5±9.0	135.6±9.0	135.4±10.0	135.5±9.0	135.7±10.0	135.4±9.0	135.4±9.0
S	143.3±7.0	141.7±6.0	138.8±6.0	137.8±7.0	136.8±6.0	130.3±6.0	130.2±7.0	130.3±7.0	130.2±6.0	130.3±7.0	130.2±7.0
T	144.7±11.0	149.9±11.0	144.4±12.0	146.0±14.0	144.6±11.0	136.3±11.0	136.4±11.0	136.4±11.0	136.5±11.0	136.2±11.0	136.2±11.0
U	171.0±6.0	179.0±4.0	170.7±6.0	177.7±4.0	170.3±5.0	172.7±21.0	163.0±4.0	172.9±21.0	163.0±4.0	169.1±13.0	162.9±4.0
V	146.6±14.0	142.8±14.0	142.6±15.0	141.3±14.0	140.7±12.0	134.5±13.0	134.6±13.0	134.9±13.0	134.7±13.0	135.0±13.0	134.4±13.0
W	226.8±19.0	221.5±14.0	221.5±13.0	220.4±14.0	217.4±16.0	207.1±14.0	206.0±13.0	206.8±14.0	206.2±13.0	208.3±17.0	206.0±13.0
X	149.7±11.0	152.4±12.0	149.2±12.0	149.2±12.0	147.8±10.0	139.8±12.0	140.0±12.0	140.0±12.0	139.9±12.0	139.6±12.0	139.6±12.0
Y	147.3±4.0	150.3±6.0	147.3±4.0	146.2±5.0	144.9±6.0	136.7±5.0	137.0±4.0	136.6±4.0	136.9±4.0	136.7±5.0	136.6±5.0
Z	172.6±11.0	172.8±11.0	170.8±9.0	171.6±8.0	167.1±9.0	157.7±9.0	157.8±9.0	157.8±9.0	157.9±9.0	157.8±9.0	157.9±9.0
0	54.1±2.0	51.6±3.0	51.6±3.0	53.7±2.0	51.6±3.0	48.1±3.0	48.2±3.0	48.4±3.0	48.0±3.0	48.0±3.0	47.9±3.0
1	46.6±2.0	45.6±2.0	45.6±2.0	49.4±2.0	45.8±2.0	42.8±2.0	42.9±1.0	42.7±1.0	42.8±1.0	43.0±1.0	42.6±2.0
2	122.1±8.0	114.1±7.0	115.1±7.0	116.7±7.0	113.6±8.0	107.9±6.0	107.7±7.0	107.8±6.0	107.9±7.0	107.8±6.0	107.6±6.0
3	122.7±3.0	112.2±4.0	110.8±4.0	114.1±2.0	111.4±4.0	105.5±3.0	105.8±3.0	105.6±3.0	105.5±3.0	105.7±3.0	105.5±3.0
4	147.0±12.0	145.1±11.0	145.1±11.0	144.0±11.0	143.6±13.0	134.9±11.0	135.0±11.0	135.1±11.0	135.0±11.0	134.8±11.0	135.4±11.0
5	155.4±6.0	144.4±6.0	144.9±5.0	145.8±4.0	143.5±5.0	134.9±4.0	135.3±4.0	135.4±4.0	135.1±4.0	135.1±4.0	135.1±4.0
6	87.8±3.0	83.0±2.0	82.5±2.0	87.5±2.0	82.4±2.0	76.7±2.0	77.2±2.0	77.1±2.0	77.1±2.0	76.9±2.0	77.0±2.0
7	102.4±5.0	98.7±5.0	96.7±6.0	101.4±4.0	97.5±5.0	91.3±4.0	91.5±4.0	91.7±4.0	91.3±5.0	91.4±4.0	91.3±5.0
8	103.5±7.0	102.8±8.0	99.5±9.0	101.8±7.0	100.0±8.0	92.8±7.0	93.1±8.0	92.9±8.0	92.9±8.0	93.1±7.0	93.0±7.0
9	85.5±5.0	83.8±4.0	83.2±5.0	84.4±4.0	81.5±5.0	77.0±4.0	77.1±5.0	76.9±4.0	77.0±4.0	76.8±5.0	76.8±5.0
Average	138.3	137.4	135.1	135.5	133.4	126.2	125.9	126.3	125.9	126.3	125.8
σ	44.0	44.1	43.5	42.2	42.4	40.6	40.1	40.6	40.2	40.7	40.2

Table 4: Average distance from the approximated median to each string in the set. (Copenhagen Chromosomes set)

Class	Set Median	Mollineda ε	Mollineda S^M	Fischer ε	Fischer S^M	JR-S ε Freq × Cost	JR-S S^M Freq × Cost	JR-S ε Freq	JR-S S^M Freq	Hinarejos S^G	Hinarejos S^M
cromo1	47.8±3.0	49.8±3.0	44.8±2.0	57.6±4.0	44.6±2.0	42.4±3.0	42.0±3.0	42.2±3.0	42.1±3.0	42.1±3.0	42.0±3.0
cromo2	42.8±1.0	47.5±3.0	40.5±2.0	65.3±5.0	40.8±2.0	37.7±2.0	37.9±2.0	37.8±2.0	37.7±2.0	38.0±2.0	37.8±2.0
cromo3	38.7±1.0	45.4±6.0	36.4±0.5	50.4±5.0	36.2±1.0	34.2±1.0	34.2±1.0	34.3±1.0	34.3±1.0	34.5±1.0	34.5±1.0
cromo4	36.2±1.0	37.0±2.0	33.1±1.0	52.6±1.0	33.6±1.0	31.8±1.0	31.9±1.0	31.9±1.0	31.8±1.0	31.8±1.0	31.8±1.0
cromo5	33.1±1.0	37.1±0.3	30.7±1.0	51.8±3.0	31.1±2.0	28.9±1.0	28.8±1.0	28.8±1.0	28.8±1.0	28.8±1.0	28.7±1.0
cromo6	33.8±2.0	36.7±1.0	32.3±2.0	46.5±7.0	30.5±1.0	29.8±1.0	29.8±1.0	29.9±1.0	29.8±1.0	29.8±1.0	29.8±1.0
cromo7	29.5±1.0	37.0±3.0	27.1±0.2	42.2±5.0	27.2±1.0	25.9±1.0	25.9±1.0	25.9±1.0	25.9±1.0	25.9±1.0	25.9±1.0
cromo8	27.6±1.0	32.1±1.0	24.8±0.1	44.7±1.0	25.1±1.0	24.0±1.0	23.9±1.0	23.9±1.0	23.8±1.0	23.9±1.0	23.9±1.0
cromo9	28.2±2.0	28.2±1.0	28.2±2.0	28.2±3.0	28.2±2.0	28.2±1.0	28.2±1.0	28.2±1.0	28.2±1.0	28.2±1.0	28.2±1.0
cromo10	25.9±1.0	30.2±1.0	25.5±1.0	39.5±6.0	24.9±1.0	23.2±1.0	23.2±1.0	23.2±1.0	23.2±1.0	23.3±1.0	23.2±1.0
cromo11	24.6±2.0	26.4±5.0	21.0±2.0	33.3±5.0	22.3±2.0	21.3±2.0	21.3±2.0	21.3±2.0	21.3±2.0	21.2±2.0	21.3±2.0
cromo12	25.4±1.0	27.0±1.0	24.5±1.0	33.8±4.0	23.9±1.0	22.6±1.0	22.7±1.0	22.6±0.5	22.7±0.5	22.6±0.5	22.8±0.5
cromo13	20.4±1.0	24.9±3.0	19.5±1.0	35.2±3.0	18.7±1.0	18.2±1.0	18.1±1.0	18.1±1.0	18.1±1.0	18.0±1.0	18.1±0.5
cromo14	23.2±1.0	22.0±0.5	21.8±0.5	31.1±5.0	21.4±0.4	20.5±1.0	20.5±1.0	20.6±1.0	20.6±1.0	20.5±1.0	20.6±1.0
cromo15	21.0±1.0	23.4±2.0	19.4±1.0	30.7±2.0	19.8±1.0	18.6±1.0	18.6±1.0	18.6±1.0	18.6±1.0	18.6±1.0	18.6±1.0
cromo16	17.9±0.5	18.0±3.0	16.3±0.3	29.4±2.0	16.6±0.5	15.9±1.0	15.9±1.0	15.9±1.0	15.9±1.0	15.9±1.0	15.9±1.0
cromo17	21.3±1.0	22.5±1.0	19.7±2.0	30.2±3.0	19.8±1.0	18.6±1.0	18.6±1.0	18.6±1.0	18.6±1.0	18.6±1.0	18.6±1.0
cromo18	18.5±1.0	21.5±2.0	17.8±1.0	26.1±2.0	16.9±1.0	16.2±1.0	16.3±1.0	16.1±1.0	16.1±1.0	16.1±1.0	16.1±1.0
cromo19	12.4±1.0	16.2±3.0	11.6±1.0	19.5±2.0	12.0±1.0	11.7±1.0	11.7±1.0	11.7±1.0	11.7±1.0	11.7±1.0	11.7±1.0
cromo20	15.2±2.0	21.3±1.0	14.9±2.0	24.9±1.0	14.6±2.0	14.1±1.0	14.1±1.0	14.1±1.0	14.1±1.0	14.1±1.0	14.1±1.0
cromo21	10.6±1.0	12.7±2.0	10.5±1.0	14.7±3.0	10.2±1.0	10.0±1.0	10.0±1.0	10.0±1.0	10.0±1.0	10.0±1.0	10.0±1.0
cromo22	13.4±0.3	15.9±2.0	13.1±0.4	23.1±1.0	12.7±0.2	12.4±0.3	12.4±0.2	12.4±0.3	12.4±0.3	12.3±0.3	12.4±0.3
Average	26.6	30.0	24.9	38.4	24.9	23.7	23.7	23.7	23.7	23.7	23.7
σ	9.9	10.5	9.2	13.5	9.2	8.6	8.6	8.6	8.6	8.6	8.6

Table 5: Comparison of the average distance from the approximated median to each string in the set respect the true median. (Synthetic data)

Set	Exact Median	Set Median	Mollineda ε	Mollineda S^M	Fischer ε	Fischer S^M	JR-S ε Freq $\times$ Cost	JR-S S^M Freq $\times$ Cost	JR-S ε Freq	JR-S S^M Freq	Hinarejos S^G	Hinarejos S^M
Synthetic 1	6.5	6.9	7.7	6.8	6.9	6.9	6.5	6.5	6.8	6.5	6.5	6.5
Synthetic 2	7.9	8.4	8.6	8.3	8.4	8.4	8.1	8.1	8.1	8.1	8.1	8.1
Synthetic 3	8.0	8.5	8.3	8.4	8.5	8.5	8.2	8.0	8.0	8.0	8.0	8.0
Synthetic 4	7.3	7.6	7.6	7.6	7.6	7.6	7.3	7.3	7.4	7.3	7.3	7.3
Average	7.4	7.8	8.0	7.8	7.9	7.8	7.5	7.4	7.6	7.4	7.5	7.5
σ	0.6	0.7	0.4	0.6	0.7	0.7	0.7	0.7	0.5	0.7	0.7	0.7

References

Bunke, H., Jiang, X., Abegglen, K., Kandel, A., 2002. On the weighted mean of a pair of strings. Pattern Analysis & Applications 5, pp. 23-30.

Casacuberta, F., Antonio, M., 1997. A greedy algorithm for computing approximate median strings. In: VII Simposium Nacional de Reconocimiento de Formas y Análisis de Imágenes (AERFAI), pp. 193-198.

Fischer, I., Zell, A., 2000. String averages and self-organizing map for strings. In: Proceedings of the Neural Computation, Canada / Switzerland, (ICSC). Academic Press, pp. 208-215.

Freeman, H., 1974. Computer processing of line-drawing data. Computer Surveys 6, pp. 57-96.

García-Díez, S., Fouss, F., Shimbo, M., Saerens, M., 2011. A sum-over-paths extension of edit distances accounting for all sequence alignments. Pattern Recognition,Vol 44(6), pp. 1172-1182.

González-Rubio, J., Casacuberta, F. 2010. On the use of median string for multi-source translation. In: 20th International Conference on Pattern Recognition (ICPR), pp. 4328-4331.

Jain, A., Zongker, D., 1997. Representation and recognition of handwritten digits using deformable templates. IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. 19, pp. 1386-1390.

Jiang, X., Bunke, H., 2002. Optimal lower bound for generalized median problems in metric spaces. Structural, Syntactic, and Statistical Pattern Recognition, LNCS, Vol. 2396, pp. 143-151.

Jiang, X., Schiffmann, L., Bunke, H., 2000. Computation of median shapes. In: 4th Asian Conf. on Computer Vision, pp. 300-305.

Jiang, X., Bunke, H.,Csirik, J., 2004. Median Strings: A Review Data Mining in Time Series Databases. World Scientific, Vol. 57, pp. 173-192

Jiang, X., Wentker, J., Ferrer, M., 2012. Generalized median string computation by means of string embedding in vector spaces. Pattern Recognition Letters, Vol. 33, pp. 842-852.

Kohonen, T., 1985. Median strings. Pattern Recognition Letters 3, pp. 309–313.

Kohonen, T., 1998. Self-organizing maps of symbols strings. Neurocomputing, Vol. 21, pp. 19-30.

Kruskal, J., 1983. An overview of sequence comparison. time warps, string edits and macromolecules. SIAM Reviews, Vol. 2, pp. 201-2037.

Kruzslicz, F., 1999. Improved greedy algorithm for computing approximate median strings. Acta Cybernetica, Vol. 14, pp. 331-339.

Levenshtein, V. I., 1966. Binary codes capable of correcting deletions, insertions, and reversals. Tech. Rep. Vol. 8.

Lourenço, A., Fred, A., 2005. Ensemble methods in the clustering of string patterns. In: 7th IEEE Workshops on Application of Computer Vision (WACV/MOTIONS), Vol. 1, pp. 143-148.

Martínez-Hinarejos, C., Juan, A., Casacuberta, F., 2003. Median strings for k-nearest neighbour classification. Pattern Recognition Letters, Vol. 24(1-3), pp. 173-181.

Martínez-Hinarejos, C. D., 2003. La cadena media y su aplicación en reconocimiento de formas. Ph.D. thesis.

Martínez-Hinarejos, C., Juan, A., Casacuberta, F., Mollineda, Ramón 2002. Reducing the computational cost of computing approximated median strings. Lecture Notes in Artificial Intelligence, SSPR&SPR 2396, pp. 47-55.

Martínez-Hinarejos, C. D., Alfons, J., Casacuberta, F., 2001. Improving classification using median string and NN rules. In: IX Spanish Symposium on Pattern Recognition and Image Analysis. Benicàssim. Spain, (AERFAI), Vol. 2, pp. 307-314.

Mollineda, R. A., 2004. A learning model for multiple-prototype classification of strings. In: 17th Int. Conf. on Pattern Recognition (ICPR). Vol. 4, pp. 420-423.

Nicolas, F., Rivals, E., 2005. Hardness results for the center and median string problems under the weighted and unweighted edit distances. Journal of Discrete Algorithms 3 (2-4), 390–415, combinatorial Pattern Matching (CPM) Special Issue. The 14th annual Symposium on combinatorial Pattern Matching.

Olivares, C., Oncina, J., 2008. A stochastic approach to median string computation. Lecture Notes in Computer Science, SSPR&SPR 5342, 431–440.

Rico-Juan, J., Micó, L., 2003. Some results about the use of tree/string edit distances in a nearest neighbour classification task. Pattern Recognition and Image Analysis, LNCS, Vol. 2652, pp. 821-828.

Rico-Juan, J. R., Iñesta, J. M. I., 2012. New rank methods for reducing the size of the training set using the nearest neighbor rule. Pattern Recognition Letters, Vol. 33(5), pp. 654 - 660.

Ristad, E., Yianilos, P., 1998. Learning string-edit distance. IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. 20, pp. 522-532.

Wagner, R., Fischer, M., 1974. The string-to-string correction problem. Journal of the ACM, Vol. 21, pp. 168-173.