# INCREASING RADIUS SEARCH SCHEMES FOR THE MOST SIMILAR STRINGS ON THE BURKHARD-KELLER TREE

AUTHORS: **SANTANA, O.; PEREZ, J.; RODRIGUEZ, J.C..**
Department of Informatics and Systems.
Polytechnic University of Canaries.
P.O. Box 550. Las Palmas. Canary Islands. Spain.

This work is on the line for the optimization of search schemes for the most similar strings to one. The concept of similarity is on the sense of the **DD**, directional distance of Levenshtein [LE66].

The strings dictionary is structured as a DIT_organized Burkhard_Keller tree, **BK_DIT+DD**, [BK73],[NK82] and [SP88].

The object of this work is to study the search schemes in which the search radius is initialized to the minimum reachable value, that evolves in an increasing manner; in comparison with search schemes in which the search radius is initialized to the search final radius upper bound, [SP88].

Some construction approaches for the **BK_DIT** structure to improve the search performance are proposed.

The test results are studied comparing the different approaches and search schemes themselves and the work conclusions are presented.

# INCREASING RADIUS SEARCH SCHEMES FOR THE MOST SIMILAR STRINGS ON THE BURKHARD-KELLER TREE

AUTHORS: **SANTANA, O.; PEREZ, J.; RODRIGUEZ, J.C..**
Department of Informatics and Systems.
Polytechnic University of Canaries.
P.O. Box 550. Las Palmas. Canary Islands. Spain.

**ABSTRACT:**

In this work search schemes are proposed for the most similar strings to a given one, on the sense of the Levenshtein directional distance, working on a Burkhard_Keller structure, [BK73] and [NK82], organized by the transposition_invariant distance, [SD87], using a increasing search radius as opposed to the decreasing search radius schemes, [SP88]. Some organization approaches are studied to find the best way to improve search performance. The test results are analyzed, comparing these approaches and the different search schemes.

## 0.- INTRODUCTION

This work is on the line for the optimization of search schemes for the most similar strings to one. The main contribution is the description and study of search schemes on which the search radius ——in contrast with the classic decreasing evolution—— take a line of increase modification.

On the search schemes related with this work, the dictionary is organized as a Burkhard_Keller tree, **BK**. On this tree the distance used for its construction is the transposition_invariant distance, **DIT**, building a **BK_DIT**. The dictionary strings will be stored in a tree, such that in every node ——in principle at random—— a string **C**° is chosen, and all the strings in the current subdictionary whose DIT with **C**° is **i** are linked at the branch **i**.

According to the search radii evolution, the search schemes of this work may be classified as:

Decreasing:
* Severe: **D1** for short (two phases).
* Non severe: **DA** for short (two phases).

Increasing:
* Severe: **C1** for short (one phase).
* Non severe: **CA** for short (three phases).

Section 1 presents the distances used in this work. Section 2 is concerned with decreasing radius search schemes. On section 3 a search scheme on which the search radius increases from zero to a final value is proposed. The goal is to accelerate the search radius final value convergence; on section 4 a binary approach is proposed. Section 5 introduces organization approaches to improve the performance of the search schemes. Finally, the test results and conclusions of this work are presented on the section 6.

## 1.- INVOLVED DISTANCES

Let **X** be a string from an alphabet **{α1,..,αm}** and **X<i>** the character in the position **i** of **X**; let **X<i:j>** be the character sequence from **X<i>** to **X<j>**, both included, so that if **i>j** then **X<i:j> =** μ, the null string. **|X|** is the length of **X**.

An editing operation is a pair **(δ,Ω)<>(μ,μ)**, where δ and Ω are strings with length less or equal than one, i. e., they are μ or they are a single character. String **Y** results from the application of **(δ,Ω)** on **X** if **X=σδτ** and **Y=σΩτ**; that is written **X -> Y**.

Wagner, [WF74], considers three kinds of editing operations. **(δ,Ω)** is a substitution operation if **δ<>μ**, **Ω<>μ** y **δ<>Ω**; it is an extraction operation if **Ω=μ**; and an insertion operation if **δ=μ**.

Let **S** be a sequence $S_1,S_2,...,S_n$ of editing operations. A strings sequence $X_0,X_1,...,X_n$, being $X=X_0$, $Y=X_n$ and $X_{j-1}$ **->** $X_j$ through $S_j$ for every **j=1,...,n**, is a S_derivation from **X** to **Y**.

**S** converts **X** to **Y** if there is a S_derivation from **X** to **Y**.

Let Γ be an arbitrary cost function that assigns a positive real value **Γ(δ,Ω)** to every editing operation **(δ,Ω)**. Γ can be extended to the sequence **S**:

$$\Gamma(S) = 0 \text{ if } n=0$$

$$\Gamma(S) = \sum_{j=1}^{n} \Gamma(S_j) \quad \text{if } n \geq 1 ;$$

### 1.1.- DIRECTIONAL DISTANCE

The minimum cost of the sequences transforming **X** to **Y** is named editing directional distance from **X** to **Y**, **DD(X,Y)**. The algorithm describing its computation, [UK83], [UK85] and [LA87], has been shown by Santana, Pérez an others, [SP88]. It assumes, as same as in this work, the criterion that the cost of any editing operation is equal to one.

### 1.2.- TRANSPOSITION-INVARIANT DISTANCE

$$DIT(X,Y) = \frac{1}{2} \left\lceil \sum abs(X\alpha i - Y\alpha i) + abs(|X|-|Y|) \right\rceil$$

Where **X** and **Y** are strings, **Xαi** and **Yαi** are, respectively, the appearance frequencies of the character **αi** in **X** and **Y**.

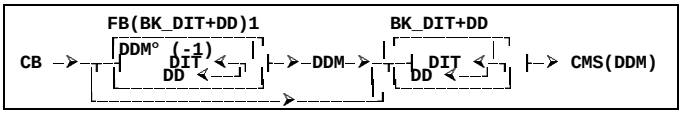
It has been proved, [SD87], that: **DIT(X,Y)** ≤ **DD(X,Y)**.

## 2.- DECREASING SCHEMES

The search procedure for the **BK_DIT+DD** scheme [SP88] is as follows. For every visited node, the branch **i** ——i.e., the **DIT** from **CB** (search string) to the string in this node—— is explored, then every branch **j** on increasing separations to **i** until a distance equal to **DDM** is explored; as in the classic joint cutoff criterion for the most similar key search on the **BK** tree. The **DDM** value begins on a upper bound of the minimum **DDM** from **CB** to the dictionary strings, **DDM°**. So, **DIT** is useful as an adaptable filter to reduce **DD** computations. **CMS(DDM)** is the set of dictionary strings whose **DD**s to **CB** are equal to **DDM**.

### 2.1.- **D1** SCHEME:

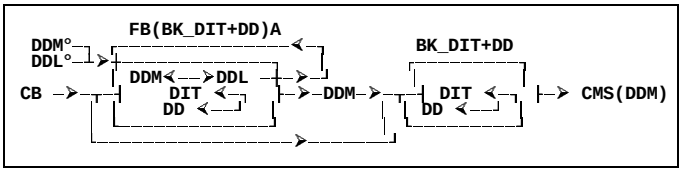
This scheme has two phases, [SP88]. The purpose of the first phase is to find out the minimum **DD** from the search string to any dictionary string. The same procedure of **BK_DIT+DD** search is used, modified so that when a **DD** less or equal than the current radius **DDM** is found, the search goes on with a radius **DDM=DD-1**. The second phase is the **BK_DIT+DD** search using the minimum **DDM** resulting from the first phase.




**D1 scheme**

2.2.- **DA** SCHEME:

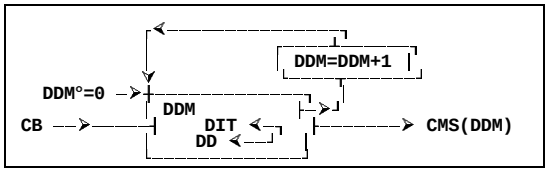
This search scheme results from a modification of **D1**'s first phase. Instead of trying to find an answer for **DD-1**, the search is performed for the average value between the current minimum **DD** and the upper value that has no answer. The aim is to reach faster the final minimum search radius, [SP88].




**DA scheme**

## 3.- SEVERE INCREASING SEARCH SCHEME, C1

On this search scheme the structure is covered beginning from the root node and a search radius **DDM=0**. While there is no answer the radius, **DDM**, is increased by one and the search begins again from the root node, using the **DIT** and **DD** already computed. In the end the answer will be the most similar strings set and their **DD** distance to **CB** is **DDM**.

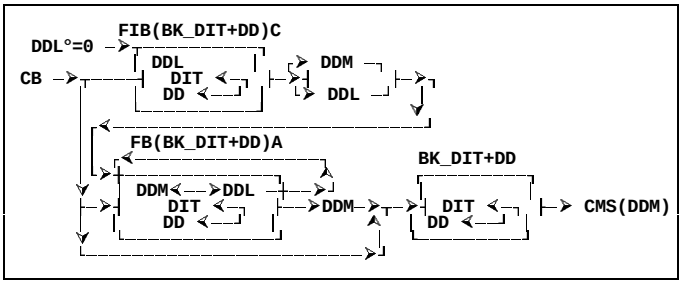



**C1 scheme**

This search scheme obtains the **DIT** filter optimum performance on the sense that the number of computed **DD**s is the minimum, using **DIT** as the only filter. This circumstance is reached for the decreasing schemes during their second phases, but having an additional cost at the first phases in order to find out the minimum **DDM**.

## 4.- NON SEVERE INCREASING SEARCH SCHEME, CA

This scheme is divided in three phases. On the first one, the structure is covered with a search radius, **DDL**, so that beginning from 0 and increasing by one, until a string gets a **DIT** to the search string less or equal than **DDL**. The **DD** computed at this point is used to initialize the upper bound of **DDM** for the second phase. The second and third phases are equal to the **DA** scheme.




**CA scheme**

## 5.- BK-DIT TREE STRUCTURAL ORGANIZATION APPROACHES

The severe increasing search scheme only computes the **DD**s from the search string to all the dictionary strings whose **DIT**s are less or equal than **DDM** (the final search radius). Therefore the computed **DD**s are independent from the dictionary distribution through the **BK** tree and so, on this search scheme, from the viewpoint of the distance computations, the unique available optimization of the organization is to decrease the number of computed **DIT**s.

An upper bound to the number of nodes covered on the severe increase search scheme is (being **h** the height of the **BK** tree): **(2*DDM + 1)**h**

If **h** decreases then this upper bound decreases, too. An effective way to decrease **h** is to increase the string distribution dispersion at the **DIT**_branches of every node.

### 5.1.- LONGEST NODE APPROACH:

At every node this approach computes, for every string in the node subfile, the maximum **DIT**, **R**, and the minimum **DIT**, **F**, to the other strings in this subfile. Then the string whose difference between their **R** and **F** is maximum is chosen. This technique will be called the **Max(R-F)** approach. It is hoped that this tactic will yield a smaller average height due to a wider **DIT**_branches distribution in every node.

### 5.2.- MAXIMUM DISPERSION APPROACH:

The aim of this approach is to choose, at every node, the string that provides the maximum **DIT**_branch distribution dispersion. So the string whose sum of the squares of the number of strings linked to every **DIT**_branches is minimum is chosen. This approach will be called **MS** approach. It is hoped that this approach gets less average height and greater number of leaf nodes than the **Max(R-F)** approach.

## 6.- TEST RESULTS AND CONCLUSIONS

A dictionary with 2089 strings was used for testing. Tests were made for different distorted strings groups generated by a distorter process from the correct dictionary. For every group the obtained distortion rate is shown.

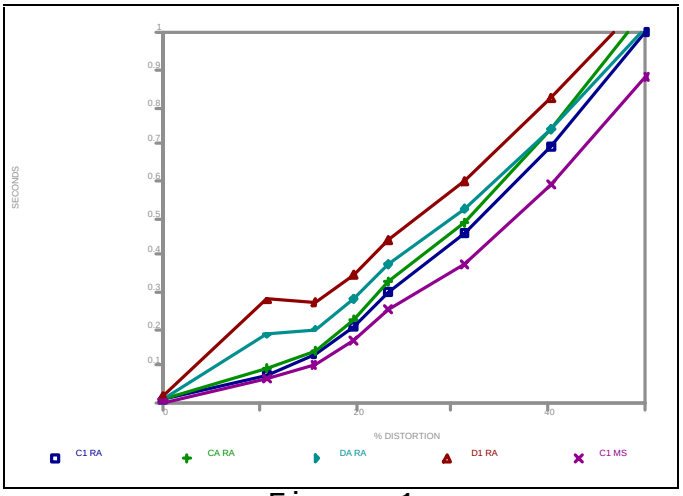



Figure 1

For low distortions the performance of the **DA** scheme, figure 1, is notably worse than on **CA** scheme, but this worsening decreases with the increase in distortion. This behaviour can be followed, figure 2, from the bringing near of

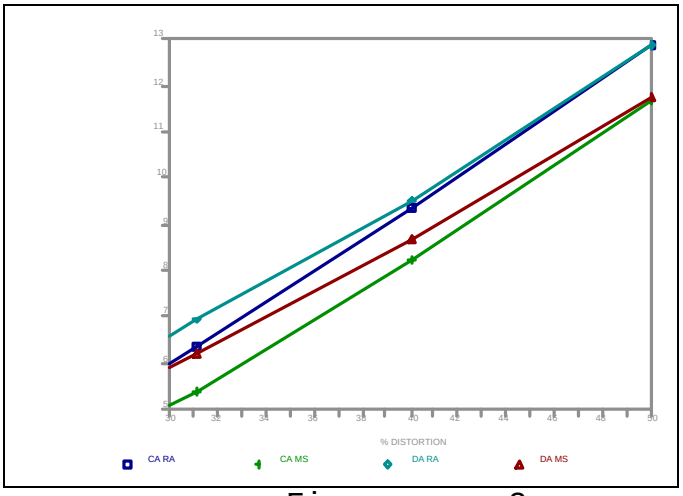



Figure 2

the **DD** equivalent cost between **CA** and **DA**. For distortions greater than 40%, figure 1, an inversion of this relative behaviour between the **CA** and **DA** schemes can be seen. This can be justified, figure 3, due to the change produced in the percentage of revisited nodes for both schemes. Since attending to the **DD** equivalent cost, figure 2, this singularity did not happen.

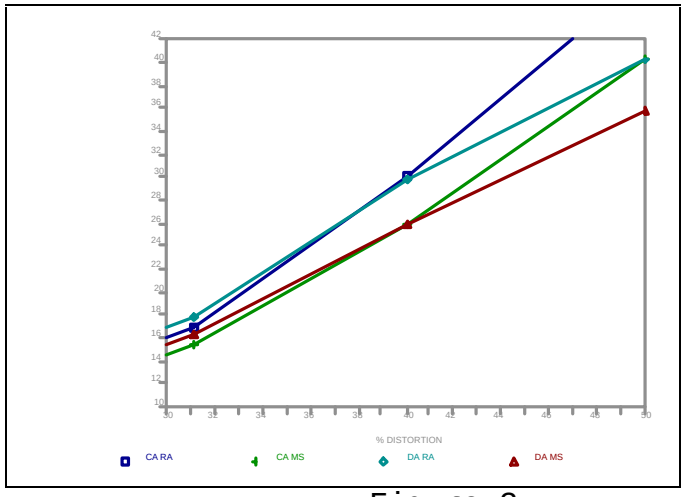



Figure 3

The use of the minimum square (**MS**) approach ——maximum string dispersion at the **DIT**_branches—— for constructing a **BK** tree improves the performance of the search schemes seen above, but the relative behaviour keeps without change, figure 4. The said inversion takes

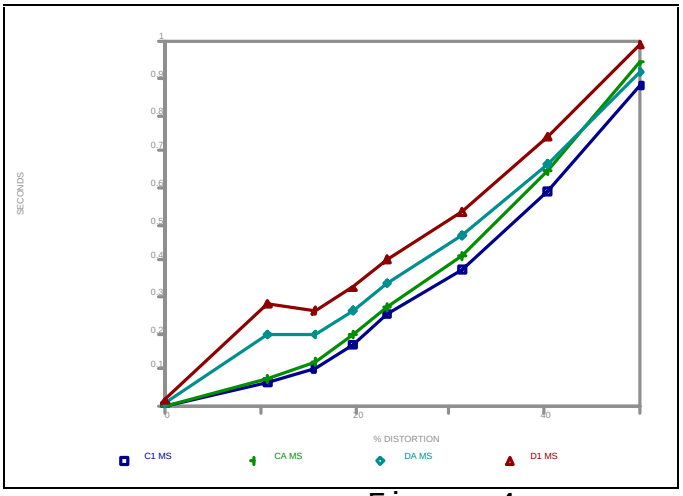



Figure 4

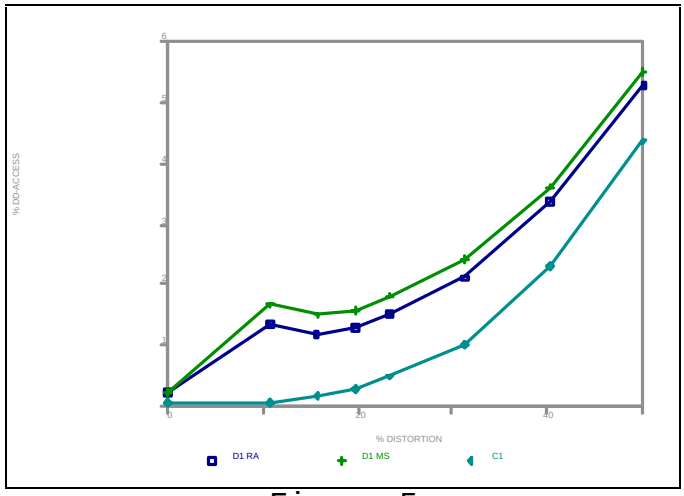



Figure 5

place, due to the same reasons, but shifted towards higher distortions, because the **DD** equivalent cost of the **DA** scheme is not so close to **CA**'s cost as on the random (**RA**) construction, figure 2, and the inversion for the revisited nodes takes place at higher distortions, figure 3.

The **C1** search scheme gets a performance notably better than **D1**, figure 1. The number of computed **DD**s on **C1** is the minimum that **DIT** does not reject, figure 5. When applying these schemes to the **MS** construction of **BK** the number of computed **DD**s for the **C1** scheme is equal to that in the **RA** construction; while for the **D1** scheme, the number of computed **DD**s increases. The number of computed **DIT**s decreases for both search schemes, **D1** and **C1**, when applying this structural organization approach ——**MS**—— being always less for the **C1** scheme than for the **D1**, figure 6. The improvement of **DIT**s computations, that **MS** approach brings, increases when the distortion increases. The difference between the **C1** search scheme and the **D1** scheme for computed **DD**s, figure 5, decreases when the distortion increases. The

**DIT** behaviour is similar to **DD**'s, figure 6.

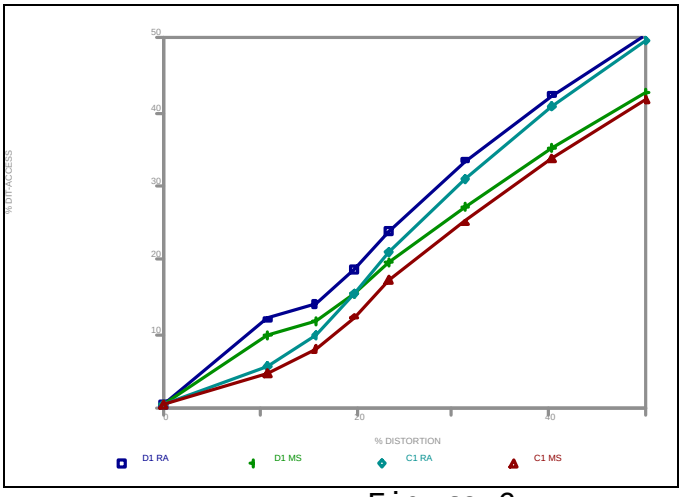



Figure 6

For all the schemes tested in this work, the performance improvement that was obtained applying the proposed construction approaches, looses efficiency when the distortion increases.

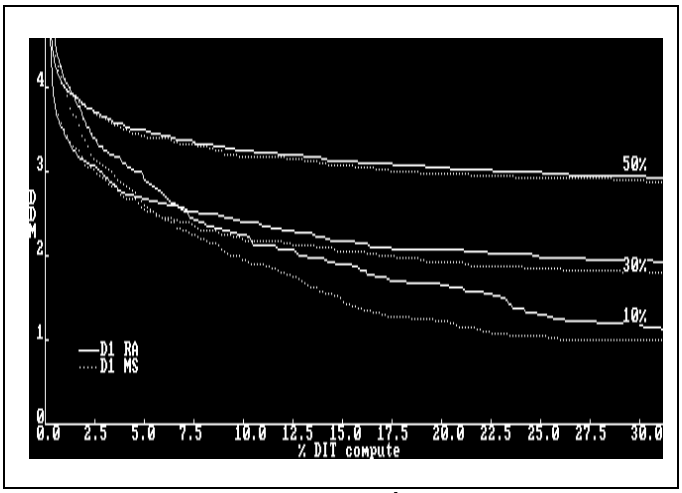



Figure 7

On figure 7, the search radius evolution on the **MS** approach shows a higher slope than on the **RA** approach. This may suggest a search performance increase, as for every **DDM** value on the **MS** construction covers a smaller subdictionary than on the **RA** construction. But this improvement is dampened, figure 8,

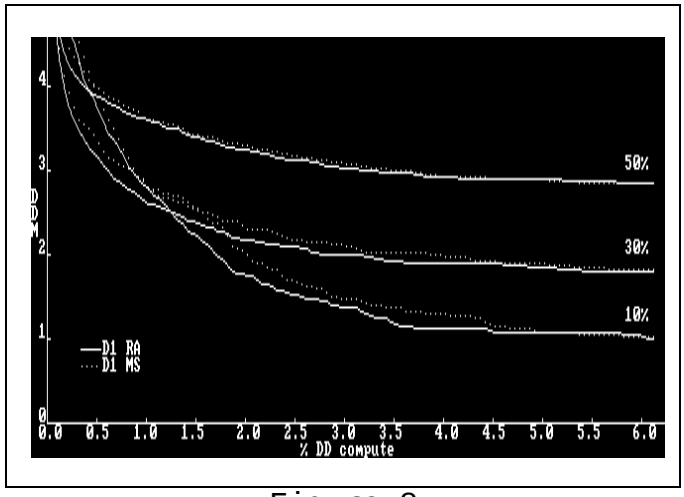



Figure 8

due to the **DDM** decrease delay with regard to the number of computed **DD**s on the **MS** construction scheme in comparison with the **RA** scheme; what brings an increase of computed **DD**s on this process. This increase is the result of that the **DIT** values —which must be computed during the coverage of the structure in the search process— are generally, figure 9, smaller on the search

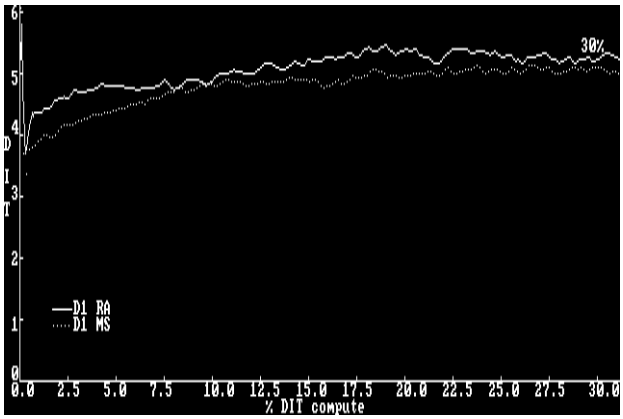



Figure 9

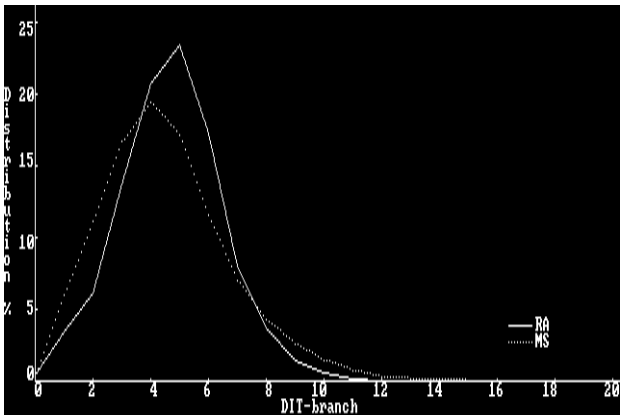



Figure 10

process on the **MS** construction than on the **RA** construction. This causes an increase in the number of **DD**_candidate strings because there is a greater easiness to pass the **DIT/DD** adaptive filter and thereby a greater number of useless **DD** computations, that do not produce a decrease in **DDM**.

The search processes are influenced by factors that, being outside the structural organization, might affect the distribution of the **DIT** values between every search string and the dictionary strings. However, this factors have not an important influence since the smaller values obtained on the search processes for the **MS** approach in relation with those obtained for the **RA** approach, figure 9, are influenced by the fact that on the **MS** approach the **DIT**_branch distribution is perceptibly shifted towards smaller distances in comparison with the **RA** organization, figure 10.

On search schemes in which **DD** must be computed for strings whose **DIT** is greater than the minimum reachable **DD** value, i.e. in which the interaction **DIT/DD** is adaptive, the use of the maximum **DIT**_dispersion approach, in comparison with the random approach, yields a smaller number of computed **DIT**s, but increases the number of computed **DD**s, and this provokes an increase, but not enough, in the general performance. It is interesting to research other approaches that attempt to reduce both numbers of computed distances, not only one. But on the search scheme **C1** the construction approach oriented to a best **DIT** organization gives the best performance.

**REFERENCES**:

**[BK73]** BURKHARD, W.A.; KELLER, R.M.: "Some Approaches to Best-Match File Searching". Comm. ACM., 16, 4, (1973).

**[LA87]** LANDAU, G. M.: "String matching in erroneous input". Thesis submitted for degree Ph Doctor Tel-Aviv University. Technical Report 57, (1987).

**[LE66]** LEVENSHTEIN, V.I.: "Binary codes capable of correcting, insertions and reversals". Soviet Phys.Dokl., 10, 707-710, (1966).

**[NK82]** NEVALAINEN, O.; KATAJAINEN J.: "Experiments with a Closet Point Algorithm in Hamming Space". Angewandte Informatik 5, 277-281, (1982).

**[SD87]** SANTANA, O.; DIAZ, M.; MAYOR, O.; REYES, J.: "Esquemas y estructura para la búsqueda de las palabras más similares a una dada". XIII Conferencia Latinoamericana de Informatica, Vol. II, 1169-1189, (1987).

**[SP88]** SANTANA, O.; PEREZ, J.; LOPEZ G.; RODRIGUEZ, G.: "La estructura de Burkhard-Keller en la búsqueda de las cadenas más similares a una dada". XIV Conferencia Latinoamericana de Informática (1988).

**[UK83]** UKKONEN, E.: "On Approximate String Matching". Proc. Int. Conf. Found. Comp. Theor., Lecture Notes in Computer Science 158, Springer_Verlag, 487/495, (1983).

**[UK85]** UKKONEN, E.; "Finding Approximate Pattern in Strings". J. of Algorithms, 6, 132/137, (1985).

**[WF74]** WAGNER, R.A.; FISCHER, M.J.: The String-to-String Correction Problem". JACM, 21 (1), 168-173, (1974).