SHARING THE COMPONENTS OF TRANSPOSITION-INVARIANT DISTANCE, DIT, ON DIT-ORGANIZED BURKHARD-KELLER STRUCTURE IN SEARCHES FOR BEST MATCHING STRINGS

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

ABSTRACT:

In this work various construction character/frequency information sharing structure approaches are proposed in order to optimize transposition_invariant distance evaluation, [SD87], that distance is used to construct a Burkhard-keller tree, [BK73] and [NK82], where is organized a dictionary of strings token over a characters alphabet [SP88], to achieve searchings of strings best matching one on Levenshtein sense [LE66].

0.- INTRODUCTION:

This work is on the line of optimizing search schemes for the strings best matching one and data structures standing them. Given a strings dictionary, a distance defined on the strings space and a search string (belonging or not to the dictionary) the question is to find all strings at minimum distance of the searching string.

The aim of this work is to organize the character/frequency information from the dictionary strings in such a way that it becomes possible to improve the transposition_invariant distance evaluation, using former evaluations by means of sharing the character/frequency information with different strings.

In section 1 the distance used is the directional distance, **DD**, introduced by Levenshtein, [LE66], evaluated by Wagner and Fisher, [WF74], and whose optimized evaluation algorithm is described in [SP88]. In section 2 appears a Transposition_Invariant distance, **DIT**, with a lower cost than **DD**, which is used as an adaptive filter to searchings [SD87]. **DITE+DD** search scheme [SD87] is described in section 3. Section 4 discusses **BK_DIT+DD** search scheme [SP88] achieved over a Burkhard and Keller (**BK**) tree structure, [BK73] and [NK82]. **DIT** components sharing structure is studied in section 5. In section 6 various construction approaches for that structure are proposed. Section 7 suggests an approach for the **BK** tree construction according to the sharing structure. Section 8 shows experimental results and conclusions of this work.

1.- DIRECTIONAL DISTANCE:

Let **X** be a string from an alphabet $\{\alpha_1,..,\alpha_m\}$, **X<i>** the character in the position **i** of **X**; **X<i:j>** the characters sequence from **X<i>** to **X<j>**, both included, if **i>j** then **X<i:j> =** μ, the nil string. $|X|$ is the length of **X**.

An editing operation is a pair $(ß,\Omega)=,/(\mu,\mu)$, where **ß** and Ω are strings of length less or equal than one, i. e., they are μ or they are a single character. String **Y** results of the application of $(ß,\Omega)$ on **X** if $X=\sigma ß \Phi$ and $Y=\sigma \Omega \Phi$, that is written **X->Y**.

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 number $\Gamma(ß,\Omega)$ to each editing operation **(ß,Ω)**. Γ can be extended to the sequence **S**:

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

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

Minimum cost of the sequences transforming **X** to **Y** is named editing directional distance from **X** to **Y**, **DD(X,Y)**. Algorithm describing its computation, [LA87] and [UK83], has been shown by Santana, Perez and others, [SP88], it follows, as in this work, the criterion that cost of any editing operation is equal to one.

2.- TRANSPOSITION-INVARIANT DISTANCE:

$$DIT(X,Y) = \frac{1}{2} \left(\sum_{i=1}^{m} |X_{\alpha_i} - Y_{\alpha_i}| + ||X| - |Y|| \right)$$

Were **X** and **Y** are strings, $\mathbf{X}_{\alpha_i}$ and $\mathbf{Y}_{\alpha_i}$ are respectively the appearing frequencies of the character α_i in **X** and **Y**.

It has been proved, [SD87], that: $\mathbf{DIT(X,Y)} \leq \mathbf{DD(X,Y)}$.

3.- DITE+DD SCHEME:

The dictionary is organized as a tree, [SD87], where components contributing to **DIT** evaluation are structured. It has a root node sorting by strings lengths. Hanging from it there is a tree-part where each node holds a character α and sorts by the various appearing frequencies, **f**, of that character in the respective subdictionary. When a tree-part branch can not sort more there is a chain-part which is a chained list holding the resting α/**f** pairs. Hanging from the chain-part is the **SIT**-chain that is a list of **DIT** synonyms.

Covering the root node begins on the position given by the search string length and alternatives are taken attending length nearness. Pathing through tree-part evaluates **DIT** components at the same time as the structure is covered beginning on the current node branch corresponding to the character appearance frequency in the search string, and continuing on nearer frequencies. Pathing through the chain-part is sequential. If **SIT**-chain is reached then **DD** between the search string and all elements on it is evaluated. Exploration limits on this structure are fixed by the searching radius.

4.- BK DIT+DD SCHEME:

In this scheme the dictionary is organized as a Burkhard & Keller tree, **BK**, using **DIT** as a construction distance. That is, dictionary strings will be stored in a tree, putting into each node, in principle at random, a string **C**, and hanging from branch **i** of the node all subdictionary strings that are at **DIT** distance **i** from **C**.

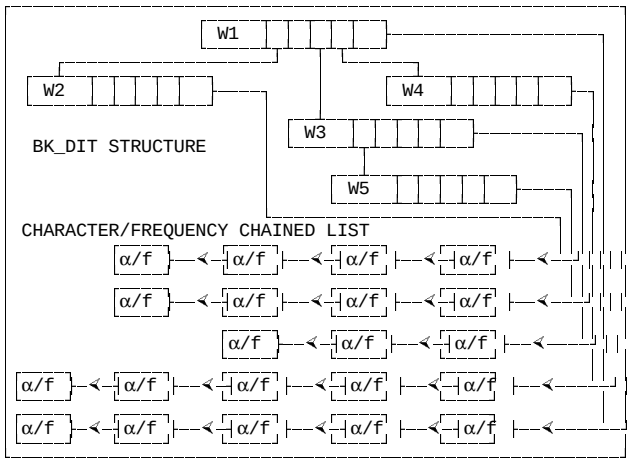


Figure 1

The search process consists of positioning the search string, **CB**, on the current node branch given by

its **DIT** to the string in that node, and continuing on all branches in a distance less or equal than **DDM** from that position, according to the usual in best match searches on **BK** trees join cut-off criterion. **DDM** value is at first equal to **DDM°** (greater than minimum **DD** between **CB** an the dictionary strings), when a **DIT** value less or equal than **DDM** is found **DD** is evaluated, if **DD** is less than **DDM** then **DDM=DD**, so in the end **DDM** will be equal to minimum **DD** between the search string and those in the dictionary. **DIT** is used as an adaptive filter to avoid **DD** evaluations. **CMS(DDM)** is the set of dictionary strings whose **DD** from **CB** is **DDM**.

A chained list of nodes, (**α/f-**nodes), hangs from each node of this **BK-**tree, (**BK-**node), as shown in figure 1. Each **α/f-**node holds information about a character and its frequency of appearance in the string joined to the **BK-**node.

In this scheme when a node is acceded the **DIT** between the search string, **CB**, and the string in the **BK-**node must be evaluated, covering all the **α/f-**node chained list. It has been shown, [SP88], that this leads to a higher evaluation of **DIT** components than that doing in the **DITE+DD** scheme, [SD87], thus loosing performance in spite of a decreasing **DD** evaluation.

The same **α/f** pair could appear in different strings, so it is possible to think of a structure which considers a sharing method using a single node for that pair.

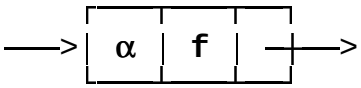
The solution proposed in this work is to reduce the number of **α/f-**nodes covered to calculate the **DIT**, by means of sharing them with different chained lists.

5.- STRUCTURING THE CHARACTER/FREQUENCY INFORMATION TO FIT SHARING:

The structure shown in figure 2 attempts, first, to reduce the number of **DIT** components to evaluate, and also, to obtain a lower load due to the **α/f-**nodes number decreasing. Number of times that both **DIT** and **DD** must be evaluated does not depend on this organization.

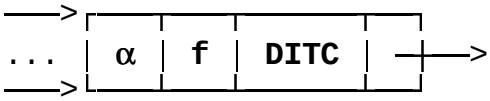
Two types of **α/f-**nodes must be distinguished: Nodes with only one predecessors (**P=1**) and nodes with a number of predecessors greater than one (**P>1**).

P=1



α : character
f : character frequency

P>1



α : character
f : character frequency
DITC: DIT component shared

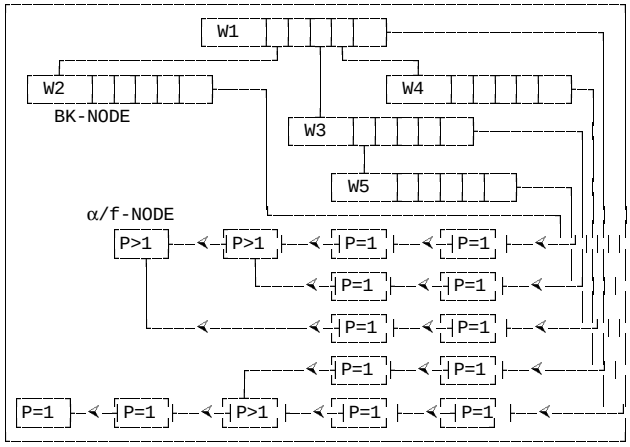


Figure 2

P=1 nodes are shared with the same set of strings that their single predecessor. **P>1** nodes are shared with the set of strings resulting from the joining of the

strings shared with all their predecessors.

In **P>1** nodes, **DITC** is of use to store the value of the **DIT** components in the **α/f-**nodes from that node to the end. Such a value will be evaluated in the first access computing the **DIT** between the search string and any of the strings sharing the node. When one of this nodes is revisited for the same search string, **DIT** between this and another of the strings sharing it is computed adding the **DITC** content to its current value.

DIT evaluation is, on the one hand, more expensive than it is on the without sharing structure, since every time a **α/f-**node is reached, it must be detected if it is of type **P=1** or **P>1**. If it is of the type **P=1** **DIT** evaluation continues in the usual way. If on the contrary it is of type **P>1** it must be tested if it has already been visited evaluating **DIT** between the search string and other strings in the dictionary; if the node has not been visited, its address and the current **DIT** value have to be recorded for storing the **DITC** value when **DIT** evaluation has been completed.

On the other hand, if **P>1** nodes are revisited then the evaluation of **DIT** components from that node to the end is saved. This effect increases according to the frequency with which nodes are revisited and the quickness to reach them in the structure.

A mechanism of any kind is needed to distinguish the first visit from later revisits to the node on **P>1** nodes. This can be solved using a search words counter.

The structure construction for information character/frequency sharing needs a research into the **α/f** pair in the dictionary strings in order to make a reasonably good sharing structure.

6.- CONSTRUCTION APPROACHES FOR THE DIT COMPONENTS SHARING STRUCTURE:

Selection of the character/frequency sorting pair in the associated space given by the dictionary strings must be oriented to construct a nodes structure grouping as possible those strings sharing **α/f** pairs, in order to reduce multiple appearance of them.

Following selection approaches are studied:

1.- Random **α/f** pair selection.

2.- Highest appearance frequency **α/f** pair selection.

3.- Random α character selection.

4.- Maximum strings dispersion α character selection.

6.1.- RANDOM α/f PAIR SELECTION:

It consists of randomly selecting an **α/f** pair from those in the dictionary strings and then dividing the strings set into two subsets, one with those strings containing the pair and another with those not containing it. An **α/f-**node holding the selected pair and pointing its predecessor will be created for the set of strings containing the pair. The process is recursively repeated with all generated subsets while they have more than one string. When a subset has only one string, **α/f** pairs still non-treated are not shared with other strings, so, as many **P=1** nodes as resting pairs will be created.

6.2.- HIGHEST APPEARANCE FREQUENCY α/f PAIR SELECTION:

Structure construction by this approach is similar to that of the above paragraph, but the selected **α/f** pair will be that contained by a highest number of strings. Since maximum sharing is attempted, in

each set that **α/f** pair shared with a highest number of strings is selected.

6.3.- RANDOM α CHARACTER SELECTION:

While more than one string will exist and characters will remain to treat, a character α is randomly selected from those remaining. As much **α/f-**nodes as different frequency values of the character α in the considered subset are created for this character, dividing such subset into new subsets containing the strings with the different α appearing frequencies. The rest of the considerations are similar to those in the above paragraph.

6.4.- MAXIMUM STRINGS DISPERSION α CHARACTER SELECTION:

Structure construction by this approach is similar to that of the above paragraph but character selection is not random. Maximum dispersion of the number of hanging from each subset branch strings distribution is attempted, since it is supposed that a higher dispersion will give rise to a higher sharing. To maximize dispersion that character α for which the dividing original subsets cardinals squares addition is minimum is selected.

7.- BK TREE CONSTRUCTION ACCORDING TO THE SHARING STRUCTURE:

Each of the proposed sharing structure construction approaches give rise to a **α/f-**nodes organization which does not depend on the dictionary strings distribution on the **BK** tree.

It seems reasonable to think that, during search process, the sharing structure performance with respect to **DIT** components evaluation will be better the closer the treated strings are to it. However, the order in which strings are treated during searching is fixed by the **BK** tree organization. So it is possible to establish a searching structure construction approach to get a better performance of **DIT** components evaluations which will lead to a better performance of the structure considered in all.

Such a construction **BK** tree approach can be expressed as: putting in each tree node the string which will share a greater number of **α/f** pairs with all the other strings hanging from it.

8.- EXPERIMENTAL RESULTS AND CONCLUSIONS:

To perform the experiments, a dictionary of 2089 strings was used. Searches were achieved for

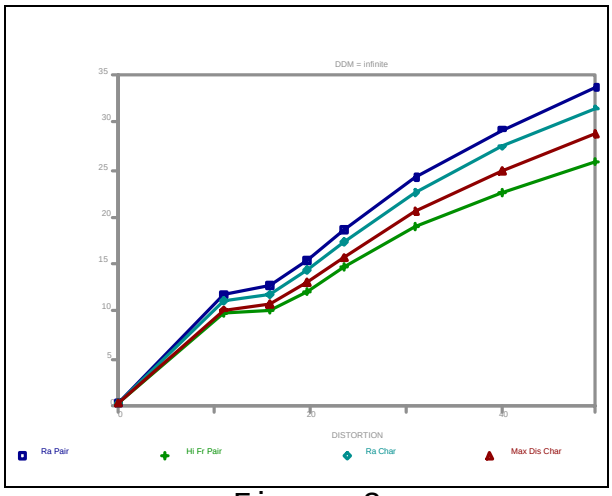


Figure 3

different sets of strings obtained from a distorting process over the good strings dictionary. The distorting grade is shown for every case.

Table 1 shows, for proposed approaches, the values of the sharing ratio, defined as follow: **100*(1-NC/NT)**, where **NC** is the number of **α/f** nodes in the structure and **NT** is the total number of non-equal characters per string in the dictionary.

This ratio measures the sharing structure occupation improvement, figure 2, against the separate lists of α/f nodes organization, figure 1.

CONSTRUCTION APPROACHES OF THE SHARING STRUCTURE	SHARING RATIO
RANDOM α/f PAIR	16.24
HIGHEST FREQUENCY α/f PAIR	30.71
RANDOM α CHARACTER	17.60
MAXIMUM STRINGS DISPERSION α CHARACTER	22.50

Table 1

Like figure 3 shows random

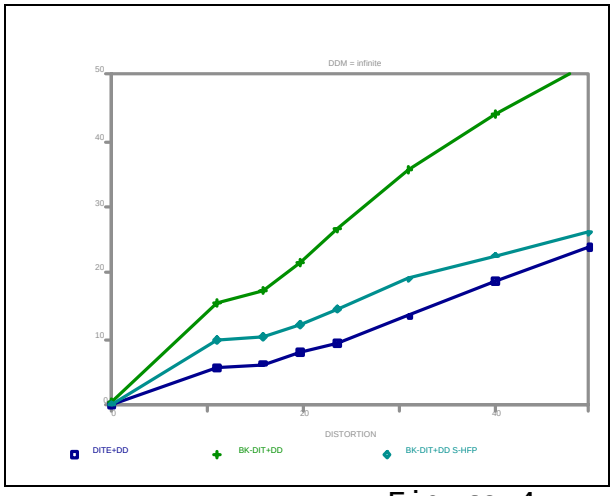


Figure 4

selection approaches of the sorting object give a worse performance than those oriented to maximize sharing. On random cases, character α selection approach presents a better behaviour than **α/f** approach.

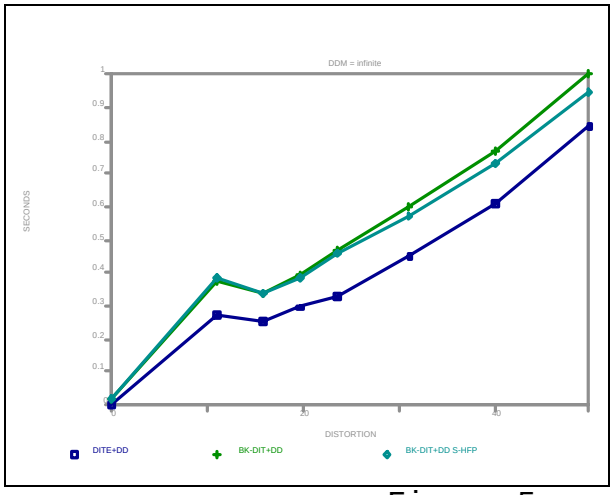


Figure 5

However, when decision is oriented to a more complex approach, a reversing occurs to this trend. So goodness of **α/f** pair approach is the greatest reached with these approaches. These behaviours are according to relationships shown in table 1. Furthermore, although with all these approaches the performance gets worse with distortion, with more frequent **α/f** pair this process is slower. So this approach is the most adequate for both load and searching processes viewpoint.

It is convenient to note that figures 3 do not represent the global performance of this schemes but only a partial view, although it is the distinctive aspect since all other tasks are similar and independent of the sharing structure construction approach.

A notable reduction of **DIT** percent with respect to **BK_DIT+DD** scheme has been obtained, figure 4, by means of sharing **DIT** components, **S-HFP**. This becomes more evident with increasing distortion, in spite of more **DIT**s always being evaluated than in **DITE+DD** scheme.

As figure 4 shows, the performance of the proposed search scheme does not approach as closely as hoped to that of **DITE+DD**, figure 5. That is due to the greater complexity of its **DIT** evaluations, as mentioned in section 5. The greater complexity of **DIT** evaluation leads to a result which, at low distortions, is slightly worse than that for **BK_DIT+DD**, but with an increase in distortion, figure 5, a better **DIT** evaluation becomes clear since use of sharing improves with the increment in the number of explored strings.

DD evaluations are not influenced by this structural change. Better results could be reached over the shown structure using multiple phases search schemes in order to reduce **DD** evaluation, [SP88].

BK tree construction according to the sharing structure (**CAS**), figure 6, shows a better performance on low distortions as regards random construction due to a better use of sharing since it follows the path through **BK** nodes for the search scheme more closely. When the number of strings to be treated grows, following distortion increase, the sharing structure must be covered more broadly, so decreasing importance of this construction approach. However, it never presents a worse performance, so it seems recommendable to construct the **BK** tree according to the highest appearing frequency **α/f** pair structure.

It is of interest to attempt the study of this construction performance against other search schemes. Multiple phases search schemes have been studied [SP88], which could be used over the structure shown in this work, and even growing radius search schemes, [SP89], instead of decreasing radius as previously, have been studied.

Finally, it must be added that all future works using **BK_DIT+DD** structure must incorporate the sharing structure shown in this work.

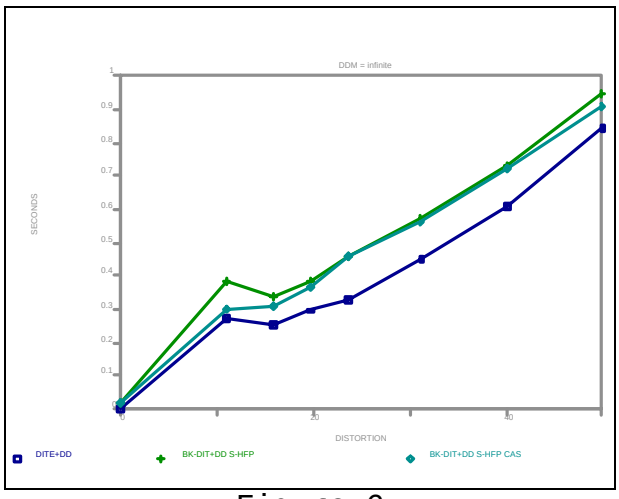


Figure 6

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).

[SP89] SANTANA, O.; PEREZ, J.; RODRIGUEZ,J.C,: "Increasing Radius Search Schemes for the Most Similar Strings on the Burkhard-Keller Tree". EUROCAST'89, (1989).

[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).