Center Selection Techniques for Metric Indexes [*]

Cristian Mendoza Alric
Depto de Informática, Universidad Nacional de San Luis
Ejército de los Andes 950, San Luis, Argentina
calric@unsl.edu.ar

and

Norma Edith Herrera
Depto de Informática, Universidad Nacional de San Luis
Ejército de los Andes 950, San Luis, Argentina
nherrera@unsl.edu.ar

ABSTRACT

The metric spaces model formalizes the similarity search concept in nontraditional databases. The goal is to build an index designed to save distance computations when answering similarity queries later.

A large class of algorithms to build the index are based on partitioning the space in zones as compact as possible. Each zone stores a representative point, called center, and a few extra data that allow to discard the entire zone at query time without measuring the actual distance between the elements of the zone and the query object. The way in which the centers are selected affects the performance of the algorithm.

In this paper, we introduce two new center selection techniques for compact partition based indexes. These techniques were evaluated using the Geometric Near-neighbor Access Tree (GNAT). We experimentally showed that they achieve good performance.

Keywords: *Databases, Metric Spaces, Similarity Search, Index, Centers Selection.*

1 INTRODUCTION

The similarity or proximity search concept is frequently found in diverse topics in computer science, i.e. voice and image recognition, text compression, computational biology, data mining, etc. In [6] is shown that the proximity search problem can be expressed as follows: given a set X of objects and a distance function d defined among them, that quantifies their similitude, the aim is to retrieve all the elements similar to a given one. This function d satisfies the properties required to be a distance function: positivity ($d(x,y) \geq 0$), simetry ($d(x,y) = d(y,x)$) and triangular inequality ($d(x,y) \leq d(x,z) + d(z,y)$). The pair (X,d) is called ***metric space***. A finite subset $U \subseteq X$, which will be called ***database***, is the set of objects where the search takes place.

One of the typical queries over this new database model is the ***range query***, denoted by $(q,r)_d$. Given a query $q \in X$ and a tolerance radius r, a range query consists in retrieving all the objects from the database U that are within a distance r from q, that is: $(q,r)_d = \{u \in \mathcal{U} : d(q,u) \leq r\}$.

The total query time T can be computed as $T =$ *#evaluations of* $d \times$ *complexity*(d) + *extra CPU time* + I/O *time*. In many applications, the evaluation of function d is so costly that the other terms in the formulae can be neglected. This is the complexity model used in this work; therefore, the complexity measure will be the number of evaluations of the distance function d.

A range query can be trivially answered by an exhaustive examination of the database. Unfortunately, this is generally very costly in real applications, ($O(n)$ distance evaluations where $n = |U|$). To avoid this situation, the database is preprocessed using an indexing algorithm whose aim is to build a data structure or index, designed to save distance evaluations at query time.

In [6] the authors present a unifier development for all the existing solutions in this topic. In that survey the authors state that the metric spaces indexing algorithms are based on, first, partitioning the space into equivalence classes and, second, a subsequent indexation of each class. Afterwards, at query time, some of these classes can be discarded using the index and an exhaustive search takes place only on the remaining classes.

The difference among the existing indexing algorithms is the way they build up the equivalence relation. Basically, two groups can be distinguished: pivot based algorithms and compact partition based algorithms. The ***pivot based algorithms*** [2, 5, 4] use

[*]This work has been supported by Project 22F/314, Universidad Nacional de San Luis

the distance between the database elements and a set of preselected elements or pivots in order to define the equivalence relationship. Using this criterion, two elements are equivalent if they are at exactly the same distance from all the pivots. The ***compact partition based algorithms*** [8, 3, 7] build the equivalence relation based on the proximity of the elements to a predefined set of them, called centers. In this sense, two elements are equivalent if the closest center to each of them is the same center c. Even though the way the the centers are selected affects the performance of the index, most of these algorithms choose them randomly.

In this work we have studied metric space indexing using compact partition based algorithms. Specifically, we have focused on the index called *Geometric Near-neighbor Access Tree (GNAT)* [3] aiming to design center selection techniques that improve its performance in answering range queries. The developed techniques were experimentally evaluated comparing their results with the naive center selection, i.e. random selection.

This paper is organized as follows: section 2 presents a brief explanation of metric spaces indexing algorithms. In section 3 the GNAT index is explained in detail. Sections 4 and 5 introduce and develop the center selection techniques proposed in this work. Finally, in section 6, the conclusions and future work are presented.

2 SIMILARITY SEARCH ALGORITHMS

As it was mentioned in the introduction, the metric spaces indexing algorithms can be classified into pivot based and compact partitions based. The following paragraphs give a brief explanation of each of them.

Pivot based algorithms. The key idea behind the pivot based algorithms is as follows. At indexing time, k pivots $p_1, p_2, \cdots, p_k$ are chosen from the metric space X and each object $u \in U$ is mapped to a k-dimensional vector which represents the respective distances to the pivots, that is, $\delta(u) = (d(u,p_1), d(u,p_2), ..., d(u,p_k))$. When a range query $(q,r)_d$ is issued, the triangular inequality is used together with the pivots in order to filter out elements in the database, without actually evaluate each distance to the query q. To do this, $\delta(q) = (d(q,p_1), d(q,p_2), ..., d(q,p_k))$ is computed; if $|d(q,p_i) - d(u,p_i)| > r$ holds for any for any p_i then, by the triangular inequality, $d(q,u) > r$ without actually evaluating $d(q,u)$. The distance from the query q to the elements not discarded by the previous condition is then evaluated to determine whether they belong or not to the answer.

Compact partitions based algorithms. In these case, the key idea is to split the space in zones as compact as possible. A set of centers $c_1, c_2, \cdots, c_k$ is chosen and each element in the space is associated to its closest center c_i. The set of the elements closer to the center c_i than to any other center forms the class $[c_i]$. There are many possible criteria to discard classes when the index is used to answer a query; the most popular ones are:

a. Hiperplane criterion: it is the most basic and the one that best expresses the idea of compact partition. Basically, if c is the center of class $[q]$ (i.e. the center closest to q) then, the ball with center q does not intersect $[c_i]$ if $d(q,c) + r < d(q,c_i) - r$.

b. Covering radius criterion: in this case, the method tries to bound the class $[c_i]$ by considering a ball centered in c_i that contains all the elements of U which belong to the class. We define the covering radius of $c \in U$ as $cr(c) = max_{\ u \in [c] \cap \mathcal{U}}\ d(c,u)$. Then $[c_i]$ can be discarded if $d(q,ci) - r > cr(c_i)$.

One of the main issues found in the design of efficient indexing techniques is what is known as the curse of dimensionality. The dimensionality concept is related to the effort needed to search for an element in a given metric space. The intrinsic dimensionality of a metric space X is defined in [6] as $\rho = \frac{\mu^2}{2\sigma^2}$, where μ and σ^2 are the mean and variance of the distance histogram of X. This states that, as the intrinsic dimensionality increases, so do the mean, but the variance of the histogram decreases, i.e. the distance histogram concentrates around its mean and this creates a negative effect on indexing algorithms.

3 GEOMETRIC NEAR-NEIGHBOR ACCESS TREE

This index, proposed by Sergey Brin in [3], is an extension of the Generalized Hiperplane Tree (GHT) [8] to an m-ary tree. The construction of a GNAT of arity m can be summarized as follows: for the first level m centers $\{c_1, \ldots, c_m\}$, are chosen from X. Then, the set U_{c_i} formed by all the objects of U closer to c_i than to any other center c_j, is related to each center c_i:

$$\mathcal{U}_{c_i} = \{x \in \mathcal{U} / d(c_i, x) < d(c_j, x),\ \forall j = 1 \ldots m,\ j \neq i\}$$

If $|U_{c_i}| > m$, a GNAT is recursively created as a child of node c_i, otherwise a terminal node is built using the elements of U_{c_i}.

The GNAT stores at each node an $O(m^2)$ size table $\rho_{ij} = [\min_{u \in \mathbb{U}_j}(c_i,u), \max_{u \in \mathbb{U}_j}(c_i,u)]$, that keeps minimum and maximum distances from each center c_i to each set U_{c_j}. At query time, this information is used together with the triangular inequality, to limit the search. Given a range query $(q,r)_d$, q is compared against some center c_i. Then, any other center c_j (and their corresponding sets U_{c_j}) such that $[d(q,c_i) - r, d(q,c_i) + r]$ does not intersect $\rho_{i,j}$ is discarded (see figure 1) . This process is repeated until no center can be discarded. Then, the search recursively continues in the remaining subtrees. During

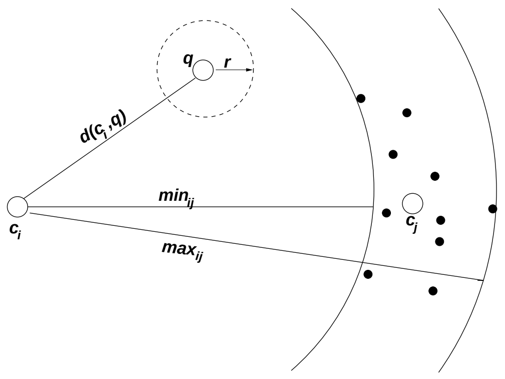


Figure 1: Pruning branches using ρ_{ij}. In this case, $\mathcal{U}_{c_j}$ can be discarded since $d(q, c_i) + r < min_{x \in \mathcal{U}_{c_j}} d(c_i, x)$.

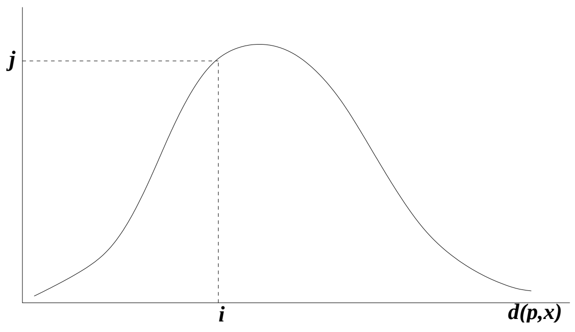


Figure 2: Example of the local histogram with respect to p. Here, $j = |\{x/d(p,x) = i\}|$
.

this process, all the centers c_i that satisfy $d(c_i, q) \leq r$ are added to the result.

The arity chosen for the construction of the GNAT strongly influences its performance. In some metric spaces, a high arity can be a good choice, while for some others, a low arity can result in better performance [1]. Furthermore, the technique used to select the centers during the construction of the GNAT also affects its efficiency at query time. This is the reason because in this work we have focused in the study and design of center selection techniques that improve the index performance at query time.

4 CENTER SELECTION TECHNIQUES

The most sensible feature in a metric space is its geometry, i.e. how the data is distributed. The knowledge of this underlying structure of the dataset is very useful in the design of indexing algorithms. More precisely, the knowledge of how they are distributed and clustered helps in identifying the zones where the search becomes harder.

One way to visualize the data distribution of the metric space is using distance histograms. Given a metric space $(\mathcal{X}, d)$ and an element $p \in \mathcal{X}$, the local histogram respect to the reference point p, is the distribution of distances from p to every $x \in \mathcal{X}$ (see figure 2).

In [1] the authors define and characterize the concept of hard and soft kernel of a metric space. The hard kernel is formed by those elements which lay in a densely populated zone of the metric space; on the other hand, the soft kernel is formed by the rest of the elements of the metric space. In the mentioned work, the authors also show the algorithm to calculate the hard kernel of a metric space. That algorithm basically consists in intersecting the middle zone of local histograms for several different reference points p.

Considering this ideas, we have designed two techniques for center selection. One of them consists in selecting the centers from elements which belong to the soft kernel and the other one does it picking elements from the hard kernel. In both cases, we did not actually calculate the kernels of the metric space through distance histogram intersection (as suggested in [1]), but we worked at each step of the selection only with the local histogram of the last chosen center. These ideas are further explained as follows:

Closer Element(*CE*). The first center c_1 is randomly chosen. The second center is then chosen from the zone that would be considered soft kernel with respect to c_1, i.e. if the histogram has Gauss bell shape, the elements would be chosen from its extremes. In order to achieve this, the local histogram of c_1 is built and the closest element in the histogram is chosen to be the second center c_2. In general, the center c_{i+1} would be the element closest to c_i.

High Density Zone (*HDZ*). In this technique also, the first center c_1 is chosen randomly. Once c_i has been selected, the center c_{i+1} is chosen from the zone that c_i would consider as its hard kernel. In order to achieve this, the local histogram for c_i is computed and an element from the central zone of the c_i local histogram is selected as the center c_{i+1}. This zone is the most densely populated in the histogram, if it is Gauss bell shaped.

According to the guideline provided in [1], the high density zone of elements can be determined using the mean in the local distance histogram for c_i. The idea is to select as center c_{i+1} an element whose distance to c_i lies in the interval $[\mu - x, \mu + x]$, where μ is the mean of the local histogram for c_i and x is an integer number. The most convenient value for x was experimentally obtained as explained in section 5.

5 EXPERIMENTAL RESULTS

The experiments were performed over word dictionaries using the edit distance (also called Leveshtein distance) as distance function. This function is discrete and computes the minimum number of character insertions, deletions and replacements needed to make

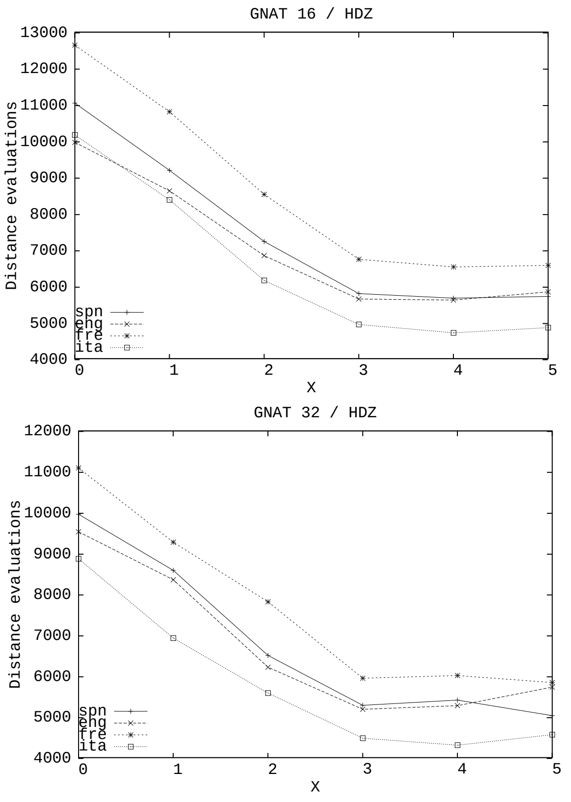


Figure 3: *Varying x in HDZ selection for arities* 16 *and* 32.

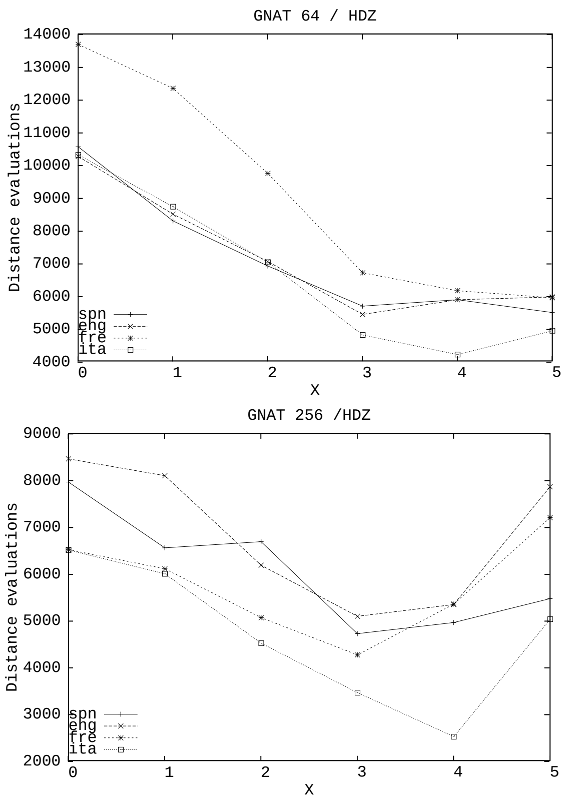


Figure 4: *Varying x in HDZ selection for arities* 64 *and* 256.

two strings equal. This distance is commonly used in information retrieval, signal processing and computational biology applications.

Four dictionaries were actually used, namely, Spanish (86.061 words), French (138.257 words), Italian ($116,879$ words) and English ($69,069$ words). The dictionaries were indexed using GNATs built with arities $2,4,8,16,32,64,128,256$ and 512.

The experiments have been split into two phases. The first phase aimed to determine the most appropriate value of x to be used for *HDZ* technique on each dictionary. Once this value was established, on phase two, the experiments were run using the two proposed techniques. In order to obtain a measure of their performance, the results of phase two were compared against the naive center selection technique: the random selection (***RND***).

Because of space reasons, in the following sections only most relevant results will be included; intermediate results will not be exposed but are available on demand.

Choosing the x value

The first set of experiments was run using the values $0,1,2,3,4$ and 5 for x aiming to determine its best value for the different GNAT arities. In this phase, the dataset to be indexed consisted of a sample of 50% of the elements, randomly selected from each dictionary. The full dictionary was not used because the aim was to obtain some guiding results for the most convenient value for x. A smaller set of words from the dictionary was chosen instead (5% of the dictionary) and the range search was performed using radius $r = 1$.

Figures 3 and 4 show the results for some arities on the Spanish, French, Italian and English dictionaries, denoted by *spn*, *fre*, *ita* and *eng*, respectively (results for the other arities are available). The x values used in these experiments are represented on the x-axis, while the y-axis represents the average number of distance evaluations done to solve a range query with radius $r = 1$. As can be observed in the figures, all the arities of the GNAT preserve a pattern that can be identified on the curves drawn for the different dictionaries. This allows us to determine the best value for x that will be used to run the experiments in phase two. From these figures, it can be determined that for each GNAT of arity less than 64 (see figure 3), the most suitable value for x can be either 3 or 4. A sudden improvement is experienced by the curves from values $x = 0$ to $x = 3$, beyond this value, they tend to stabilize. Curiously, for GNATs of arity greater than 64 (figure 4), the curves for the different dictionaries lose their similitude and decreasing tendency as the value of x increases.

Given that the plots of figures 3 and 4 were drawn only based on results for search radius $r = 1$ and over a reduced dictionary, these are only a guideline for the x value. Later, it was experimentally shown that, for some reduced set of arities of the GNAT, these values

Arity	Spn	Fre	Ita	Eng
2	$x = 4$	$x = 4$	$x = 4$	$x = 4$
4	$x = 4$	$x = 4$	$x = 4$	$x = 4$
8	$x = 4$	$x = 4$	$x = 4$	$x = 4$
16	$x = 4$	$x = 4$	$x = 4$	$x = 4$
32	$x = 3$	$x = 4$	$x = 4$	$x = 3$
64	$x = 3$	$x = 4$	$x = 4$	$x = 3$
128	$x = 4$	$x = 3$	$x = 3$	$x = 3$
256	$x = 2$	$x = 4$	$x = 2$	$x = 3$
512	$x = 3$	$x = 3$	$x = 4$	$x = 2$

Table 1: *x values for* HDZ *technique.*

did not yield the best results for all the search radii. Since the best value for x varies for different search radii and that this value is required at indexing time, it was decided to chooses the number i as value for x if i yields a performance value closest to the optimal in all the cases. Table 1 shows the values that were finally selected for each arity and for the different dictionaries.

Evaluation of the proposed techniques

In this phase, the indexes were built using 90% of the elements of each dictionary, leaving the remaining 10% as queries for range search with radius $r = 1, 2, 3$ and 4. Given that each GNAT node is size of $O(m^2)$, where m is the arity of the tree, the experiments had two main objectives. First of all, to analyze which technique yields better results under the condition of using the same amount of memory, i.e. same arity. Afterwards, to study which technique results in better overall performance, without considering the amount of memory used.

Even though the complexity measure used is the number of distance evaluations, the main memory needed has to be considered as an influent factor. If the index does not fit in main memory, the amount of time spent in I/O operations can affect the its efficiency. Because of this, a balance must be achieved between the overall performance of the index at search time and its arity, in order to avoid response time degradation due to I/O operations.

Figures 5 and 6 show the results for the different techniques on the Spanish dictionary. The GNAT arities are represented on the x-axis and, on the y-axis are the average number of distance evaluations used to answer the range query. As can be seen, for arities less or equal than 64 , all the center selection techniques have a common behavior in the sense that their graphs decrease as the arity of the GNAT increases. *HDZ* technique has shown better performance than the other techniques; for high selectivity queries ($r = 1$) *HDZ* saves around 40% of distance

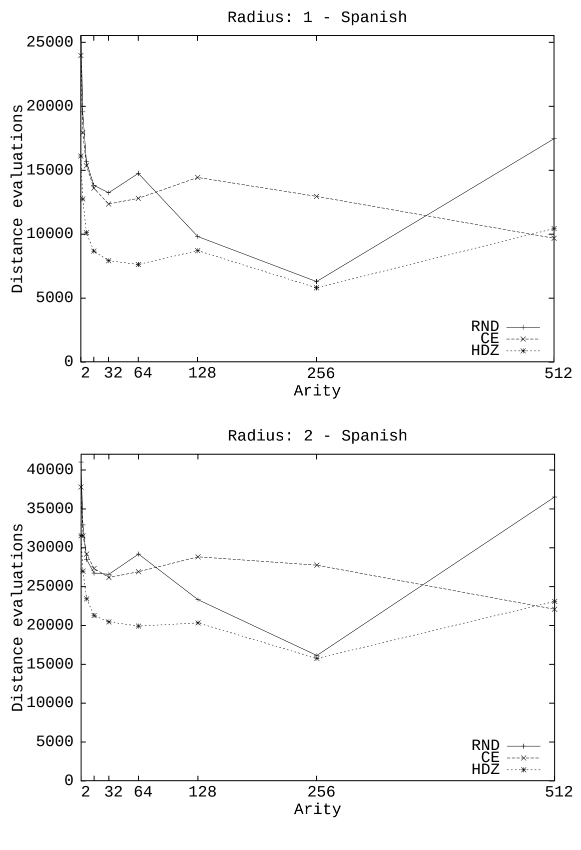


Figure 5: *Search costs for Spanish dictionary using search radii* 1 *y* 2

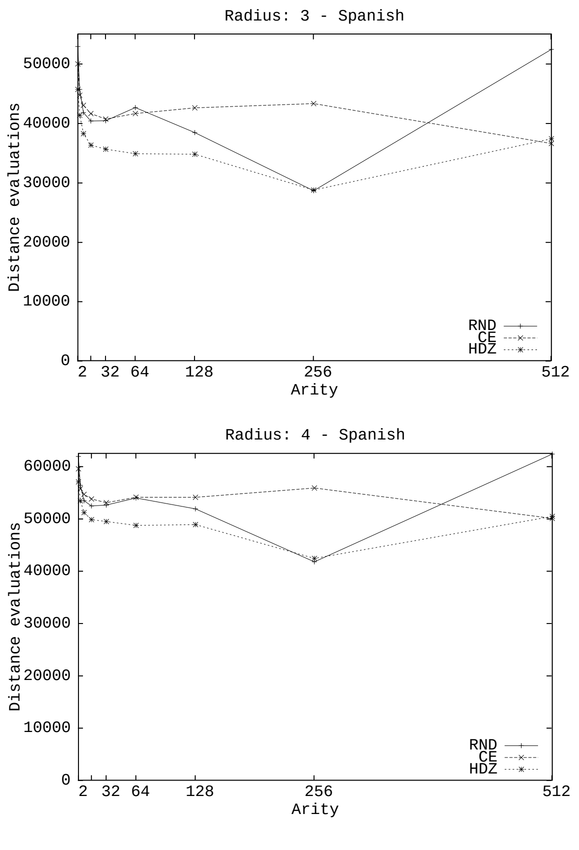


Figure 6: *Search costs for Spanish dictionary using search radii* 3 *y* 4

evaluations and for low selectivity queries ($r = 4$) the improvement has been up to 10%.

As mentioned in previous sections, GNAT with arities greater than 64 do not always help to improve the index performance at query time. In *RND* and *HDZ* techniques, the performance degrades when the arity is increased from 256 to 512 and in the *CE* technique the same phenomenon is exposed as arity grows from 64 to 128. In the later case its performance is even worse than a random selection. *RND* and *HDZ* achieve their best performance when arity 256 is used and *CE* achieves its best performance with arity 512. Note that *CE*, even using twice the memory demanded by the other policies, does not overcome their performance.

From this results, we can conclude that *HDZ* outperforms the others techniques in all the cases considered. The *RND* selection shows little performance improvement than *HDZ* for arity 256 and search radius $r = 4$. If the available memory is not enough to store the entire 256 arity GNAT, clearly the 64 arity GNAT using the *HDZ* selection, would be the best choice.

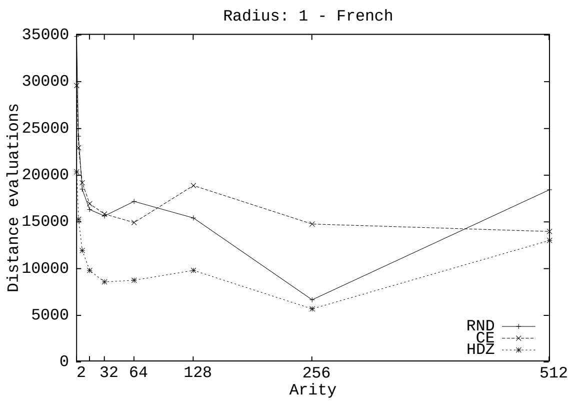
The results obtained using the metric spaces based on the rest of the dictionaries followed the same pattern as the Spanish dictionary. Figure 7 shows the results obtained when indexing metric spaces based on French, Italian and English dictionaries, respectively using radius $r = 1$ (results for other radii are available).

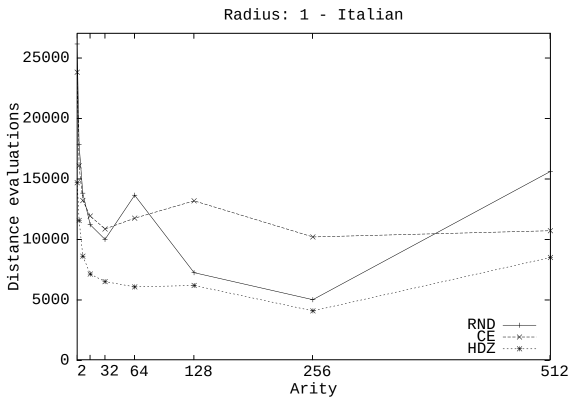
Even when the patterns shown by the graphs of the English dictionary are similar to those of the rest of the dictionaries, it is remarkable the improvement shown by *CE* over *HDZ* for arity 512, performing 10% to 20% less distance evaluations. Anyway, its lowest value does not overcome the performance shown by *HDZ* with arity 256.

Summarizing, from the results obtained, two major approaches can be considered: one of them consists in finding the balance between the average number of distance evaluations and the required memory, while the second is to minimize the average number of distance evaluations.

If the first approach is considered, the use of GNAT with arities 16, 32 or 64 together with *HDZ* technique is the most suitable choice. On the other hand, if the aim is to minimize the average number of distance evaluations (second approach), the GNAT with arity 256 using *HDZ* technique, results in more than acceptable general values; only for low selectivity searches over the Spanish and English dictionaries, it is barely bitten by the GNAT with arity 256 using random policy. The figures 8 and 9 summarize the behavior of *HDZ* technique in all the considered cases .

CE has similar performance than *RND* for Spanish and English dictionaries when using GNAT with arities lower than 32 and high selectivity queries. In general, for arities 128 and 256, its behavior is worse than *RND*, nevertheless its performance improves for





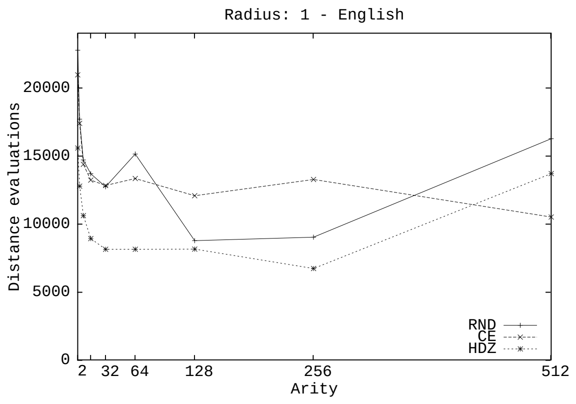


Figure 7: *Search costs for French, Italian ang English dictionaries using search radius $r = 1$.*

all search radii as arity increases beyond these values. We suspect that its main problem is the effect of choosing the closest element of the current center as the next center. The only way this technique has to incorporate diversity in the chosen centers is to increase their number, i.e. a GNAT of arity 512. The previous analysis arises this policy as a field where further development and investigation is required.

6 CONCLUSIONS AND FUTURE WORK

In this paper, two new center selection techniques to be used during GNAT building have been introduced;

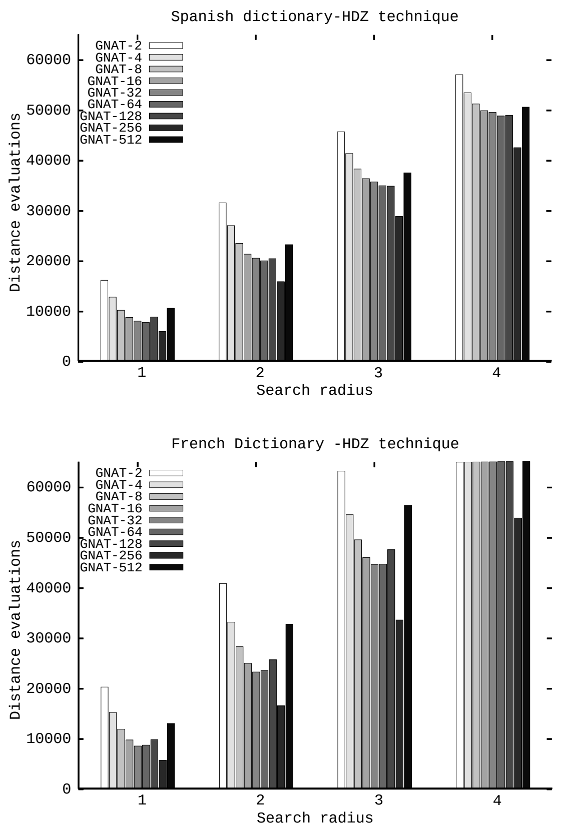


Figure 8: *Search costs for Spanish and French dictionaries using HDZ technique.*

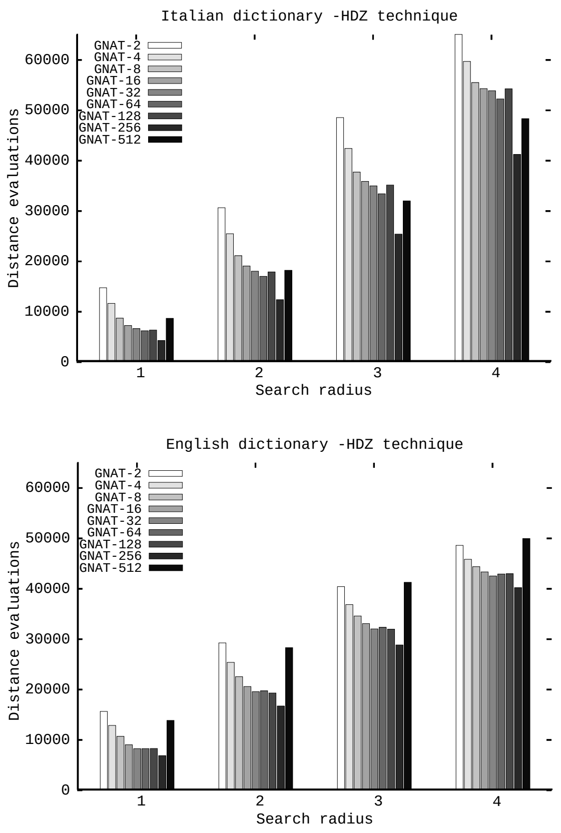


Figure 9: *Search costs for Italian and English dictionaries using HDZ technique.*

namely, closer element and high density zone. The least competitive was the *CE* selection, having even worse performance than the random selection. The most competitive was the *HDZ*, achieving important decrease in the number of distance evaluations when compared to the random selection.

Concerning the future work, we propose to study the behavior of these policies over some other metric spaces, adapting them to the new environment if necessary. Bear in mind that the presented policies are applicable to metric spaces with bell shaped distance histograms and that many metric spaces do not satisfy this condition.

We will also keep investigating the cause of the poor performance of the *CE* selection and introduce some modifications in order to improve it.

REFERENCES

[1] R. Baeza-Yates, B. Bustos, E. Ch´avez, N. Herrera, and G. Navarro. *Clustering in Metric Spaces and Its Application to Information Retrieval.* Kluwer Academic Publishers, 2003. ISBN 1-4020-7682-7.

[2] R. Baeza-Yates, W. Cunto, U. Manber, and S. Wu. Proximity matching using fixed-queries trees. In *Proc. 5th Combinatorial Pattern Matching (CPM'94)*, LNCS 807, pages 198–212, 1994.

[3] S. Brin. Near neighbor search in large metric spaces. In *Proc. 21st Conference on Very Large Databases (VLDB'95)*, pages 574–584, 1995.

[4] E. Chavez and K. Figueroa. Faster proximity searching in metric data. In *Proceedings of MICAI 2004. LNCS 2972, Springer,* Cd. de M´exico, M´exico, 2004.

[5] E. Chavez, J. Marroquin, and G. Navarro. Fixed queries array: A fast and economical data structure for proximity searching. *Multimedia Tools and Applications (MTAP)*, 14(2):113–135, 2001.

[6] E. Chavez, G. Navarro, R. Baeza-Yates, and J.L. Marroquin. Searching in metric spaces. *ACM Computing Surveys,* 33(3):273–321, September 2001.

[7] G. Navarro. Searching in metric spaces by spatial approximation. In *Proc. String Processing and Information Retrieval (SPIRE'99)*, pages 141–148. IEEE CS Press, 1999.

[8] J. Uhlmann. Satisfying general proximity/similarity queries with metric trees. *Information Processing Letters*, 40:175–179, 1991.