Adaptive and Dynamic Pivot Selection for Similarity Search

Mariano Salvetti[1], Claudia Deco[1], Nora Reyes[2], Cristina Bender[1]

[1] Facultad de Ciencias Exactas e Ingeniería, Universidad Nacional de Rosario (UNR), Rosario, Argentina
salvettimariano@hotmail.com, {deco,bender}@fceia.unr.edu.ar
[2] Departamento de Informática, Universidad Nacional de San Luis (UNSL), San Luis, Argentina
nreyes@unsl.edu.ar

Abstract. In this paper, a new indexing and similarity search method based on dynamic selection of pivots is presented. It uses Sparse Spatial Selection (SSS) for the initial selection of pivots. In order to the index suits itself to searches, we propose two new selection policies of pivots. The proposed structure automatically adjusts to the region where most of searches are made. In this way, the amount of distance computations during searching is reduced. The adjustment is done using the policy of "the best candidate" for the incoming pivot selection, and the policy of "the least discriminating" for the outgoing pivot selection.

Categories and Subject Descriptors: H. Information Systems [**H.m. Miscellaneous**]: Databases

Keywords: Metric databases, Dynamic index, Sparse Spatial Selection

1. INTRODUCTION

The digital age creates a growing interest in finding information in large repositories of unstructured data that contain textual data, multimedia, photographs, 3D objects and strings of DNA, among others. In unstructured data repositories it is more useful a similarity search than an exact search. The similarity search problem can be formalized through the concept of metric space: given a set of objects and a distance function between them, which measures how different they are, the objective is to retrieve those objects that are similar to a given one. In order to improve objects retrieval, an index can be used, because an index structure allows fast access to objects. There are several types of indexes proposed for metric spaces that have differences such as how they are explored or how they store the information.

We present a new indexing and similarity searching method based on dynamic selection of pivots. The proposed method is dynamic because it can be applied to an initially empty database that grows over time. The method is adaptive because it is not necessary to preset the number of pivots to be used because the algorithm selects pivots as necessary to self-adapt it to space complexity. To select the initial pivots of the index, the Sparse Spatial Selection (SSS) method ([Pedreira and Brisaboa 2007; Brisaboa et al. 2006]) is applied. The proposed improvement consists on implementing new policies of incoming and outgoing pivots, in order to the index suits itself to searches, to dynamic collections, and to secondary memory. Hence, we present a method that improves the SSS through the use of these two new policies.

The rest of the paper is structured as follows: Section 2 presents basic concepts and describes the problem of pivots selection. Section 3 presents the proposed method, and Section 4 shows experimental results. Finally, conclusions are presented.

2. BASIC CONCEPTS

A metric space(X, d) consists of a universe of valid objects X and a *distance function* $d : X \times X \rightarrow \mathbb{R}^+$ defined among them, which satisfies the properties: strictly positiveness $d(x, y) > 0$, symmetry $d(x, y) = d(y, x)$, reflexivity $d(x, x) = 0$ and triangular inequality $d(x, y) \leq d(x, z) + d(z, y)$. A finite subset DB of X, with $|DB| = n$, is the set of elements where searches are performed. The definition of distance function depends on the type of objects. A subclass of metric space is a *vector space*, where the most used distance function is: $L_s((x_1, \ldots, x_k), (y_1, \ldots, y_k)) = \left(\sum_{i=1}^{k} |x_i - y_i|^s\right)^{\frac{1}{s}}$. If $s = 2$, it is the Euclidean distance. The dimensionality of a vector space is the number of components of each vector. Although general metric spaces do not have an explicit dimensionality, they have an intrinsic dimensionality, following the same idea as in vector spaces. The efficiency of search methods is worse in spaces with a higher intrinsic dimensionality [Chávez et al. 2001].

In metric databases, queries of interest can be range search and k-nearest neighbors search. Given a query q and a radius r, the first one retrieves objects that are at a distance less or equal than r: $\{u \in DB / d(u, q) \leq r\}$. The second one retrieves the k objects closest to q, that is: $A \subseteq DB$ such that $|A| = k$ and $\forall u \in A, v \in DB - A, d(q, u) \leq d(q, v)$. The basic implementation is to compare each object with the query. The problem is that, in general, the evaluation of the distance function has a very high computational cost, making this search inefficient for large collections. Hence, the main goal of most search methods is to reduce the number of distance function evaluations. Building an index and using the triangular inequality, objects can be discarded without comparing them with the query. There are two types of search methods: *clustering-based* and *pivots-based* [Chávez et al. 2001]. The first one splits the metric space into a set of equivalence regions, each of them represented by a *cluster center*. During searches, whole regions are discarded depending on the distance from the cluster center to the query. *Pivot-based* algorithms select a set of objects as *pivots*. An index is built by computing distances from each database object to each pivot. Searching then computes the distances from the query q to each pivot, and discards some objects using triangular inequality and distances precomputed during the index-building phase. The papers [Chávez et al. 2001; Zezula et al. 2006; Samet 2005] have a good compilation of these search methods.

Pivots selection affects the search method efficiency in metric space. The location of each pivot with respect to the others determines the ability to exclude elements without directly comparing them with the query. Most pivots-based search methods select pivots randomly. Also, there are no guidelines to determine the optimal number of pivots, which depends on the specific collection. There are several heuristics for pivots selection. For example, pivots are objects that maximize the sum of distances among them in [Micó et al. 1994]; and several selection strategies based on an efficiency criterion to determine whether a given set of pivots is more efficient than another set of the same size in [Bustos et al. 2001]. The conclusion is that good pivots are far away from each other and from the rest of objects. In [Pedreira and Brisaboa 2007] the Sparse Spatial Selection (SSS), which dynamically selects a set of pivots well distributed throughout the metric space, is presented based based on the idea that pivots dispersed on the space discard more objects during the search. When an object is inserted into the database, it is selected as a new pivot if it is far enough from other pivots. A pivot is far enough from another if it is at a distance greater than or equal to $M \times \alpha$, in which M is the maximum distance between any two objects and α is a constant parameter that influences the number of selected pivots (with optimal experimental values around 0.4). We present an improving to Sparse Spatial Selection method, implementing new policies for selecting incoming and outgoing pivots from the index. Also the index suits itself to searches after it was adapted to the metric space. Besides, this proposal generates a number of pivots based on the intrinsic dimensionality of the space.

3. PROPOSED METHOD

We present a new indexing and similarity searching method based on dynamic selection of pivots. This method is *dynamic* because it can be applied to an initially empty database that grows over

time; and it is *adaptive* because it is not necessary to preset the number of pivots to be used since the algorithm selects pivots to self-adapt the index to the space complexity. SSS is applied to select the initial pivots of the index. Then, as time passes and searches are performed, we apply new policies for selecting pivots in order to eliminate those least discriminating pivots from the index, and to select objects as candidate pivots to put them into the index. In this way, we can adapt dynamically the index to searches performed during a given time.

3.1 Initial construction of the index and growth of the collection

Let (X, d) be a metric space, where $DB \subseteq X$ is the database. Let M be the maximum distance between objects ($M = \max\{d(x, y)|x, y \in X\}$). The value of M depends on metric space features, and in many practical cases it can be obtained without processing all objects in the collection. For example, in a vector space, M can be inferred from the maximum/minimum values of each vectors component. Also, if M is not known in advance, its value can be estimated. Although in SSS method it is not necessary to state in advance the amount of pivots to use, we set the value of α at the beginning. Parameter α is a value that depends on the features of objects. Experimentally, we conclude as [Pedreira and Brisaboa 2007] concludes that α should be between 0.35 and 0.40, depending on the dimensionality of the collection. Method efficiency is the same for all values of α in this interval. We can also see that when $\alpha > 0.40$, number of evaluations of distance functions takes higher values in spaces of high dimensionality. This is because increasing the value of α implies that the number of pivots decreases, and this has a stronger effect in spaces of higher dimensionality.

Let the collection of elements be initially empty. The first object x_1 inserted into the database is the first pivot p_1. When the second (or new) object is inserted in the database, its distance to all pivots that are already in the index is calculated. If these distances are all greater than or equal to $M \times \alpha$, this object is added to the set of pivots. Thus, the set of pivots does not have to be selected randomly because pivots are chosen as the database grows. Then, distances from the new pivot against to all database objects are calculated and stored. The next pseudocode summarizes the pivot selection process. Pivots that were selected for the initial index are far apart (over $M \times \alpha$), so all the selected pivots will not be too close to each other. Forcing the distance between two pivots to be greater or equal than $M \times \alpha$, ensures that they are well distributed in the space. For many authors, this is a desirable feature of the set of pivots.

The number of pivots depends on the initial dimensionality of the space. When the construction begins, the number of pivots should grow fast in the index, but it will be stabilized when the database grows. The following method, **InitPivots()**, also builds the index and it stores the distances between each database object and all pivots.

```
1.  InitPivots(DB:Database, d:function, M:distance, α:double) {
2.    Pivots ← {x_1}, where x_1 ∈ DB
3.    FOR ALL x_i ∈ DB DO {
4.        IF (∀p ∈ Pivots, d(x_i, p) ≥ M × α)
5.            THEN Pivots ← Pivots ∪ {x_i}
6.        END IF
7.    } END FOR ALL
8.  }
9.  Output: the Pivots set.
```

3.2 Exchange of Pivots in the Index

Given a query (q, r), the distances of q toward all pivots are calculated. By applying triangular inequality, all elements $x_i \in DB$ such that $|d(x_i, p_j) - d(p_j, q)| > r$ for any pivot p_j are discarded.

Nondiscarded objects define the list of candidates, $\{u_1, u_2, \ldots, u_l\} \subset DB$, and they should be compared directly against the query. Given an element $e \in DB$, if $\max_{1 \leq j \leq k} |d(q, p_j) - d(e, p_j)| > r$, then e is outside the query range. So, pivot p_j discriminates this object e. With searches, statistics of discrimination of each pivot and the fact that an element is part of search results are stored.

Selection policy for the Outgoing Pivot. Considering objects discriminated by several pivots and a set of B queries, we define the *percentage of discrimination* for a pivot p_i as $[\%Disc(p_i)] = \frac{Disc(p_i)}{(B \times n)}$, where $Disc(p_i)$ is the amount of items that p_i discriminates and $(B \times n)$ represents the total of possible discriminations. Then, p_i is a bad pivot when $[\%Disc(p_i)] < \frac{1}{k}$ where $\frac{1}{k}$ is an experimental threshold, which is proposed as a constant that depends on the number of pivots in the index. If $[\%Disc(p_i)] < \frac{1}{k}$, we can say that p_i is very little relevant to discriminate, at least with these B searches. Then, p_i is selected as a victim and it could be replaced in a future. After B searches the pivot with lower $[\%Disc(p_i)]$ is determined. If it is less than a threshold of tolerance with value T, it is replaced. A 10% of tolerance given by $T = \frac{1}{(1.1 \times k)}$, where k is the current number of pivots in the index, is used to stabilize the algorithm, and it was evaluated experimentally. The next pseudocode shows that after a pivot is defined as the "least discriminating pivot", it is available for exchanging using `ChangePivot()` method. The incoming pivot is provided by `GetPivot()` method. When a pivot is replaced, all distances between the incoming pivot and all elements of the database are recalculated. The complexity of changing a pivot is $n \times \theta(distance\_function)$. If discrimination percentage is not less than T, nothing is done.

```
1.  ApplySelectionPolicyOutgoing() {
2.    IF (min_{1≤j≤k} disrimine[j] < 1/(1.1×k)) THEN{
3.        OutgoingPivot ← GetPivot();
4.        ChangePivot();
5.        GenerateIndex();
6.    } END IF
7.  }
```

Selection policy for the Incoming Pivot. To choose which object becomes a pivot, the policy is to propose "the candidate pivot" using statistical data of database elements obtained from queries. If an object $e \in DB$ is frequently present in the list of candidates, we can consider that it is difficult to discriminate with the current pivots, and e will be a candidate pivot. This implies that if this element is selected as pivot, in future searches it will improve the percentage of discrimination around the region that surrounds it. Also it adapts automatically pivots to the region where most searches are made. This transforms the index into a dynamic structure, achieving its main objective: to reduce distance computations in searches. The following pseudocode shows the implementation of this policy.

```
1.  GetPivot(DB:Database, Stats:array[ ]) {
2.    Candidate ← NULL, maxCurrentStats ← 0;
3.    FOR ALL e ∈ DB DO {
4.        IF (Stats(e) > maxCurrentStats) THEN {
5.            Candidate ← e;
6.            maxCurrentStats ← Stats(e);
7.        } END IF
8.    } END FOR ALL
9.  RETURN Candidate
10. }
11. Output: element Candidate and array Stats with the time that DB
            elements are included in the list of candidates.
```

Table I. Number of pivots selected in vector spaces of dimension 8, 10, 12, 14.

dim.	n, size of the collection (x 10^3)									
	10	**20**	**30**	**40**	**50**	**60**	**70**	**80**	**90**	**100**
8	11	12	12	12	13	13	14	14	15	16
10	13	18	20	20	21	21	21	21	21	22
12	25	28	28	31	32	34	34	34	34	34
14	38	47	53	60	60	61	63	66	69	69

Table II. Efficiency in synthetic metric spaces.

Method	$dim = 8$			$dim = 10$			$dim = 12$			$dim = 14$		
	#P	**#DE**	**#DR**	**#P**	**#DE**	**#DR**	**#P**	**#DE**	**#DR**	**#P**	**#DE**	**#DR**
SSS	17	17994	6141634	24	26391	6393097	34	35721	6623490	60	62976	6915951
Proposal	15	15737	6394202	23	24380	6657667	33	34686	6717067	44	45683	7025895

4. EXPERIMENTAL RESULTS

For experimentation, two classes of metric spaces were used: synthetic and real spaces.

Synthetic Spaces: Several sets of synthetic random points in vector spaces of dimension 8, 10, 12 and 14, and Euclidean distance function are used. The database contains 100,000 objects, and range query retrieves 0.02% of database elements. Our proposal creates a dynamic amount of pivots depending on the space dimensionality, and not on the amount of database objects. For experiments, in order to achieve a uniform and distant distribution of pivots in the space, $\alpha = 0.5$ was set. This value of α was chosen from experimental results showed later (Figures 1 and 2). Table I shows the number of pivots depending on the collection size. As it is noted, the number of pivots grows very quickly with insertions of the first objects in the collection, and then continues to grow but in a slower degree until it stabilizes. So, with few elements inserted, the number of pivots depends on the number of database elements. Already with a great amount of elements inside, the set of current pivots covers all the space. Also the number of pivots in the index increases as dimension of the space increases.

To analyze the efficiency of the index for searching a database with 10,000 objects, 1,000 queries and dimensionalities 8, 10, 12 and 14, are used. 20 periods were run and information from all periods was averaged. For each dimension, the amount of pivots used in the index ($\#P$), the amount of distance functions evaluated ($\#DE$), and the amount of discriminations carried out ($\#DR$), were recorded. Results are compared against SSS because our method intends to improve it. Table II shows that the number of pivots ($\#P$) used with our proposal is always lower than in the SSS implementation, highlighting a great difference in $dim = 14$ with 16 pivots less. In the remaining dimensions, the difference is little but it remains at most 2 pivots less in our favour. This is an important result for our proposal because the pivots selection strategy of SSS presents a similar efficiency to other more complex methods and the number of pivots that it selects is close to the optimal number for other strategies. The number of evaluations ($\#DE$) for our proposal always remains below and, in general, with a uniform linear growth when the size increases. Except in $dim = 14$, where SSS shows a slight growth with the amount of reviews with a difference of about 17,000 reviews, in other dimensions the difference never exceeds 3,000. As results exposed in [Pedreira and Brisaboa 2007], the number of evaluations of the distance function in SSS is always around to the best result obtained with pivot selection techniques and strategies proposed in [Bustos et al. 2001]. So, our proposal has a number of evaluations similar to the best results obtained in previous works, even using a smaller number of pivots, which clearly implies space saving. Besides, our proposal obtains a greater number of discriminations by pivots ($\#DR$) in all dimensions. This is because, with time, the proposal makes an adjustment of pivots, and they make better discrimination reducing the amount of distance computations at query time. Thus, it shows that both selection policies of pivots are good and maintain the index dynamism.

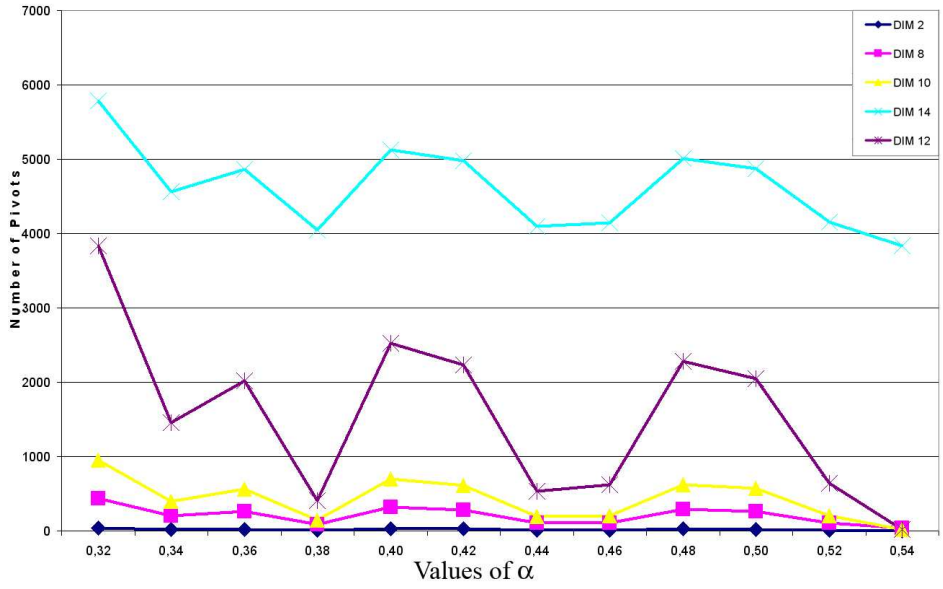

Fig. 1. Number of pivots selected by parameter α.

The value of parameter α determines the number of pivots, values between 0.35 and 0.40 are recommended, depending on the dimensionality of the space [Pedreira and Brisaboa 2007]. Here we decided to use values of α from 0.32 to 0.54, in order to evaluate the number of pivots in the index, since a rise in α represents a reduction in the number of pivots and this is noted better in spaces of higher dimension.

Figure 1 shows that for all dimensions the number of pivots varies with α, with some local maximum/minimum and large amplitude in greater dimensions. After $\alpha = 0.5$, the number of pivots decreased, as it is expected. According to these results it seems to be better to use a high α, but this is not correct since this will increase the number of distance function evaluations at re-indexing time because the index will have more pivots. Figure 2 shows the number of distance evaluations varying α. In all dimensions there is a consistent behaviour: when α increases the number of evaluations decreases. This is because when distance between pivots increases, the required distance function evaluations decrease. Uniform behaviour is because values of 20 periods were averaged: early periods have largest number of evaluations and with passing of periods, pivots were adapting themselves to searches. Also, our proposal achieves more discrimination when α increases because with passing of periods pivots are better adjusted since more elements are discriminated, so the number of distance computations decreases and searches are improved.

Real Metric Space: A database of 4000 images (photographs in black and white, people front and side), each one represented by a vector of dimension 25 and Euclidean distance function were used. A random subset of 400 images was used as queries. Values of 10 periods were averaged. The amount of pivots used ($\#P$), the amount of discriminations carried out ($\#DR$), the number of distance evaluations ($\#DE$), and the construction cost ($\#CI$) were recorded. Values of α were between 0.32 and 0.54, increasing in steps of 0.02. Results are shown in Table III.

The value of α determines the number of pivots in the index ($\#P$) and this impacts on the amount of distance function evaluations when the index is built ($\#CI$). When α increases the number of evaluations of distance function ($\#DE$) decreases fast because the amount of pivots ($\#P$) decreases. For this image space the number of pivots in the index decreases rapidly until $\alpha = 0.5$. So, we decided to use this value for α. The number of discriminations ($\#DR$) carried out by pivots decreases slowly when α increases. This behaviour validates the proposal since it is similar to results from synthetic

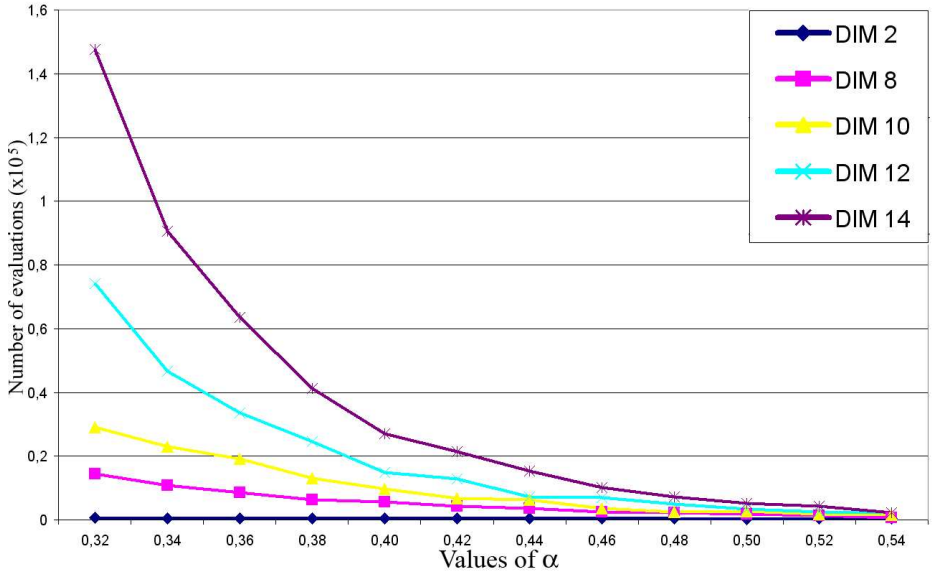

Fig. 2. Distance function evaluations according to α.

Table III. Efficiency in real metric spaces varying parameter α.

α	0.32	0.34	0.36	0.38	0.40	0.42	0.44	0.46	0.48	0.50	0.52	0.54
#P	104	73	57	42	29	27	20	15	13	10	9	7
#DR	85372	67032	65010	46418	51561	46511	49117	51371	51716	38975	37017	30549
#DE	37748	55158	56700	75011	68891	73857	71356	68854	67918	81546	79939	89456
#CI	477491	335065	258794	191035	133278	121529	89279	68271	59560	47351	42662	34119

metric spaces. Also, after $\alpha = 0.48$ the number of discriminations decreases. Unlike in the synthetic metric space, there was an increase in the number of distance evaluations ($\#DE$) when α increases.

This is because, in theory, increasing the distance between pivots should reduce distance function evaluations. This value increases because they are averaged over 10 periods of training. Also, the training set is not as representative as in synthetic metric space, because images are very different.

Figure 3 shows a query photo and the first results. Here we observe the good performance of the implementation, since in the first 3 images on the right side we get a near perfect match with a distance less than 4. This lets us know that we are making best use of the distance function between elements of the metric space. Also, we can see that the following resulting images are similar, but have more differences (e.g. the amount of hair in front of the face, the percentage of face in the image and even the orientation of the person's head).

5. CONCLUSIONS

This paper presents a new indexing and similarity search method based on a dynamic selection of pivots. One of its most important features is that it uses SSS for the initial selection of pivots because it is an adaptive strategy that chooses pivots that are well distributed in the space to achieve greater efficiency. Two new pivots selection policies are presented in order that the index suits itself to searches when it is adapted to the metric space. The proposed structure automatically adjusts to the region where most of searches are made to reduce the amount of distance computations during searches. This is done using the policy of "*the best candidate*" for the incoming pivot selection, and the policy of "*the least discriminating*" for the outgoing pivot selection. Performance of this proposal was

(a) Query photo (b) Results and distances

Fig. 3. Search results in photographs.

evaluated in synthetic and real spaces. Results show that our proposal has a number of evaluations better or similar to the best results obtained in previous works, even using a smaller number of pivots, which clearly implies space saving. We can conclude that in our proposal the index makes a greater number of discriminations at search time by a better use of historical information from previous searches with the two policies set out, with respect to SSS. Also, both selection policies of pivots maintain index dynamism. As future work we plan to implement algorithms specially designed for secondary memory, considering not only the number of distance evaluations but also the number of I/O operations. An improvement to selection policies would be to use a data warehouse for training the index with historical search data. An interesting idea to work is trying to identify nested metric spaces [Pedreira and Brisaboa 2007] and apply the proposal presented here in each of them, using different values for parameter α at an early stage to identify subspaces with high values of α in order to obtain a small amount of pivots, and a second stage implementing our proposal in each subspace.

REFERENCES

Brisaboa, N. R., Farina, A., Pedreira, O., and Reyes, N. Similarity search using sparse pivots for efficient multimedia information retrieval. In *Proceedings of IEEE International Symposium on Multimedia*. San Diego, CA, USA, pp. 881–888, 2006.

Bustos, B., Navarro, G., and Chavez, E. Pivot selection techniques for proximity searching in metric spaces. In *Proceedings of International Conference of the Chilean Computer Science Society*. Punta Arenas, Chile, pp. 33 –40, 2001.

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

Micó, M. L., Oncina, J., and Vidal, E. A new version of the nearest-neighbour approximating and eliminating search algorithm (aesa) with linear preprocessing time and memory requirements. *Pattern Recognition Letters* 15 (1): 9–17, January, 1994.

Pedreira, O. and Brisaboa, N. R. Spatial selection of sparse pivots for similarity search in metric spaces. In *Proceedings of Conference on Current Trends in Theory and Practice of Computer Science*. Harrachov, Czech Republic, pp. 434–445, 2007.

Samet, H. *Foundations of Multidimensional and Metric Data Structures (The Morgan Kaufmann Series in Computer Graphics and Geometric Modeling)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2005.

Zezula, P., Amato, G., Dohnal, V., and Batko, M. *Similarity Search - The Metric Space Approach*. Vol. 32. Springer, 2006.