Pivot Selection Techniques for Proximity Searching in Metric Spaces *

Benjamin Bustos Gonzalo Navarro
Depto. de Ciencias de la Computación
Universidad de Chile
Blanco Encalada 2120, Santiago, Chile
{bebustos,gnavarro}@dcc.uchile.cl

Edgar Chávez
Escuela de Ciencias Físico-Matemáticas
Universidad Michoacana, Edificio "B"
Ciudad Universitaria, Morelia, Mich. México
elchavez@fismat.umich.mx

Abstract

With a few exceptions, proximity search algorithms in metric spaces based on the use of pivots *select them at random among the elements of the metric space. However, it is well known that the way in which the pivots are selected can affect the performance of the algorithm. Between two sets of pivots of the same size, better chosen pivots can reduce the search time. Alternatively, a better chosen small set of pivots (requiring less space) can yield the same efficiency as a larger, randomly chosen, set. We propose an efficiency measure to compare two pivot sets, combined with an optimization technique that allows selecting good sets of pivots. We obtain abundant empirical evidence showing that our technique is effective. We also show that good pivots are outliers, but that selecting outliers does not ensure that good pivots are selected.*

1. Introduction

Many computational applications use proximity searching in a vast number of fields, for example: multimedia databases, machine learning and classification, image quantization and compression, text retrieval, computational biology, function prediction, etc.

All those applications have in common that the elements of the database form a *metric space* [8], that is, it is possible to define a positive real-valued function d among the elements, called *distance* or *metric*, that satisfies the properties of *strict positiveness* ($d(x,y) = 0 \Leftrightarrow x = y$), *symmetry* ($d(x,y) = d(y,x)$), and *triangle inequality* ($d(x,z) \leq d(x,y) + d(y,z)$). For example, a *vector space* is a particular metric space, where the elements are tuples of real numbers and the distance function belongs to the L_s family, defined as $L_s((x_1,\ldots,x_k),(y_1,\ldots,y_k)) = \left(\sum_{1\leq i\leq k} |x_i - y_i|^s\right)^{1/s}$. L_1 is the *Manhattan distance*, L_2 is the *Euclidean distance* and $L_\infty = \max_{1\leq i\leq k} |x_i - y_i|$ is called the *maximum distance*.

In general, the distance d is considered expensive to compute. Think, for example, of a biometric device that computes the distance between two fingerprints.

One of the typical queries that can be posed to retrieve similar objects from a database is a *range query* (see Section 2). An easy way to answer range queries is to make an exhaustive search on the database, but this turns out to be too expensive for real-world applications.

Proximity search algorithms build an *index* of the database and perform range queries using this index, avoiding the exhaustive search. Many of these algorithms are based on the use of *pivots*, which are distinguished elements from the database. These pivots are used, together with the triangle inequality, to filter out elements of the database without measuring their actual distance to the query, hence saving distance computations while answering the range query.

Almost all proximity search algorithms based on pivots choose them randomly among the elements of the database. However, it is well known that the way pivots are selected dramatically affects the search performance [10, 8, 9]. Some heuristics to choose the pivots better than at random have been presented [12, 4], but in general these heuristics only work in specific metric spaces and have a bad behavior in others. In $\mathbb{R}^k$ with the Euclidean metric, it is shown in [9] that it is possible to find an optimal set of $k+1$ pivots selecting them as the vertices of a sufficiently large regular k-dimensional simplex containing all the elements of the database [9], but this result does not apply to general metric spaces.

In this paper we present an efficiency criterion to

*This work has been partially supported by FONDECYT Grant 1-000929.

compare two pivot sets, which is based on the distance distribution of the metric space. Then, we present a selection technique based on this criterion to select a good set of pivots. We show empirically that this technique effectively selects good sets of pivots in a variety of synthetic and real-world metric spaces. Also, we show that good pivots have the characterisitc to be outliers, that is, good pivots are elements far away from each other and from the rest of the elements of the database, but an outlier does not always have the property of being a good pivot.

Our technique is the first we are aware of in producing consistently good results in a wide variety of cases and in being based on a formal theory.

2. Basic proximity search algorithm using pivots

There are many proximity search algorithms in metric spaces that are based in the use of pivots, such as *Burkhard-Keller Tree* (*BKT*) [5], *Fixed-Queries Tree* (*FQT*) [2], *Fixed-Height FQT* (*FHQT*) [2], *Fixed Queries Array* (*FQA*) [7], *Vantage Point Tree* (*VPT*) [12], *Multi Vantage Point Tree* (*MVPT*) [3], *Excluded Middle Vantage Point Forest* (*VPF*) [13], *Approximating Eliminating Search Algorithm* (*AESA*) [11], *Linear AESA* (*LAESA*) [10] and *Spaghettis* [6].

All these algorithms use, directly or indirectly, the following procedure to answer range queries: if the universe of objects is denoted by X, then the database is a finite subset of objects $U \subseteq X$. Given a metric space (U, d) (where d is the metric defined on U), an object $q \in X$, called the *query*, and a tolerance range $r > 0$, $r \in \mathbb{R}$, a *range query* is defined as the elements in U that are whitin distance r to q, that is:

$$(q, r) = \{u \in U,\, d(u, q) \leq r\}$$

Figure 1 shows an example of a range query in a vector space of dimension 2.

Given a range query (q, r) and a set of k pivots $\{p_1, \ldots, p_k\}$, $p_i \in U$, by the triangle inequality it follows for any $x \in X$ that $d(p_i, x) \leq d(p_i, q) + d(q, x)$, and also that $d(p_i, q) \leq d(p_i, x) + d(x, q)$. From both inequalities it follows that a lower bound on $d(q, x)$ is $d(q, x) \geq |d(p_i, x) - d(p_i, q)|$. The elements $u \in U$ of interest are those that satisfy $d(q, u) \leq r$, so we can exclude all the elements that satisfy the *exclusion condition*:

$$|d(p_i, u) - d(p_i, q)| > r \text{ for some pivot } p_i \qquad (1)$$

without actually evaluating $d(q, u)$.

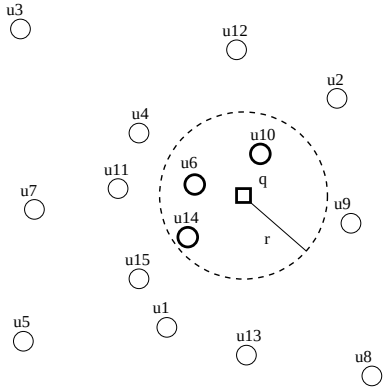

Figure 1. Example of range query, where $(q, r) = \{u_6, u_{10}, u_{14}\}$

If the space U has n elements, then the index consists of the kn distances $d(u, p_i)$ between every element and every pivot. Therefore, at query time it is necessary to compute the k distances between the pivots and the query q in order to apply the exclusion condition (1). Those distance calculations are known as the *internal complexity* of the algorithm, and this complexity is fixed if there is a fixed number of pivots.

The list of elements $\{u_1, \ldots, u_m\} \subseteq U$ that cannot be excluded by the exclusion condition (1), known as the *element candidate list*, must be checked directly against the query. Those distance calculations $d(u_i, q)$ are known as the *external complexity* of the algorithm.

The total complexity of the search algorithm is the sum of the internal and external complexity, $k + m$. Since one increases and the other decreases with k, it follows that there is an optimum k^* that depends on the tolerance range of the query. In practice, however, k^* is so large that one cannot store the k^*n distances, and the index simply uses as many pivots as space permits.

3. Efficiency criterion

Depending on how pivots are selected, they can filter out less or more elements. We define in this section a criterion to tell which from two pivot sets is expected to filter out more and hence reduce the number of distance evaluations carried out during a range query. Since the internal complexity is fixed, only the external complexity can be reduced, and this is achieved by making the candidate element list as short as possible.

Let (U, d) be a metric space. A set of k pivots $\{p_1, p_2, \ldots, p_k\}$, $p_i \in U$, defines a space P of distance tuples between pivots and elements from U. The map-

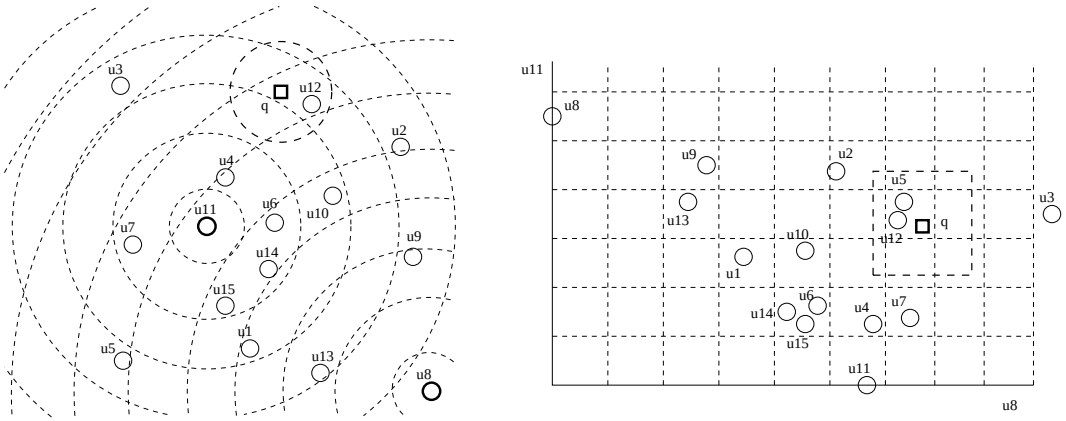

Figure 2. Mapping from a metric space onto a vector space under L_∞ metric, using two pivots.

ping of an element $u \in U$ to P, which will be denoted $[u]$, is carried out in the following way:

$$[u] = (d(u, p_1), d(u, p_2), \ldots, d(u, p_k))$$

Defining the metric $D = D_{\{p_1,\ldots,p_k\}}$ as $D([x],[y]) = \max_{1\leq i\leq k} |d(x,p_i) - d(y,p_i)|$, it follows that (P, D) is a metric space, which turns out to be $(\mathbb{R}^k, L_\infty)$. Given a range query (q, r), the exclusion condition (1) in the original space U becomes:

$$D_{\{p_1,\ldots,p_k\}}([q],[u]) > r \qquad (2)$$

for the new metric space (P, D). Figure 2 shows the mapping of the elements and the new exclusion condition. To achieve a candidate element list as short as possible, the probability of (2) should be as high as possible. One way to do this is to maximize the mean of the distance distribution of D, which will be denoted μ_D.

Another way to maximize the probability of the exclusion condition is minimizing the variance of the distribution of D at the same time, but in practice this method did not work as well as just maximizing μ_D. Hence, we will say that $\{p_1, \ldots, p_k\}$ is a better set of pivots than $\{p'_1, \ldots, p'_k\}$ when:

$$\mu_{D_{\{p_1,\ldots,p_k\}}} > \mu_{D_{\{p'_1,\ldots,p'_k\}}} \qquad (3)$$

Estimation of μ_D: An estimation of the value of μ_D is obtained in the following way:

- A pairs of elements $\{(a_1, a'_1), (a_2, a'_2), \ldots, (a_A, a'_A)\}$ from U are chosen at random.
- All the pairs of elements are mapped to space P, obtaining the set $\{D_1, D_2, \ldots, D_A\}$ of distances D between every pair of elements.
- The value of μ_D is estimated as $\mu_D = \frac{1}{A}\sum_{1\leq i\leq A} D_i$.

It is easy to see that $2k$ distance evaluations are needed to compute the distance D for each pair of elements if there are k pivots. Therefore, $2kA$ distance evaluations are needed to estimate μ_D.

4. Pivot selection techniques

Now we present three pivot selection techniques based on the efficiency criterion (3). Each technique has a cost measured in number of distance computations at index construction time. As we do more work in optimizing the pivots, better pivots are obtained. When comparing two techniques, we give them the same amount of work to spend. We describe the optimization cost of each technique.

These selection techniques can be directly adapted to work with algorithms that use a fixed number of pivots, such as *FHQT* [2], *FQA* [7], *LAESA* [10] and *Spaghettis* [6]. They can also be adapted, with modifications, to the other pivot based algorithms.

4.1. Selection of N random groups

N groups of k pivots are chosen at random among the elements of the database, and μ_D is calculated for each of this groups of pivots. The group that has the maximum μ_D value is selected.

Optimization cost: Since the value of μ_D is estimated N times, the total optimization cost is $2kAN$ distance evaluations.

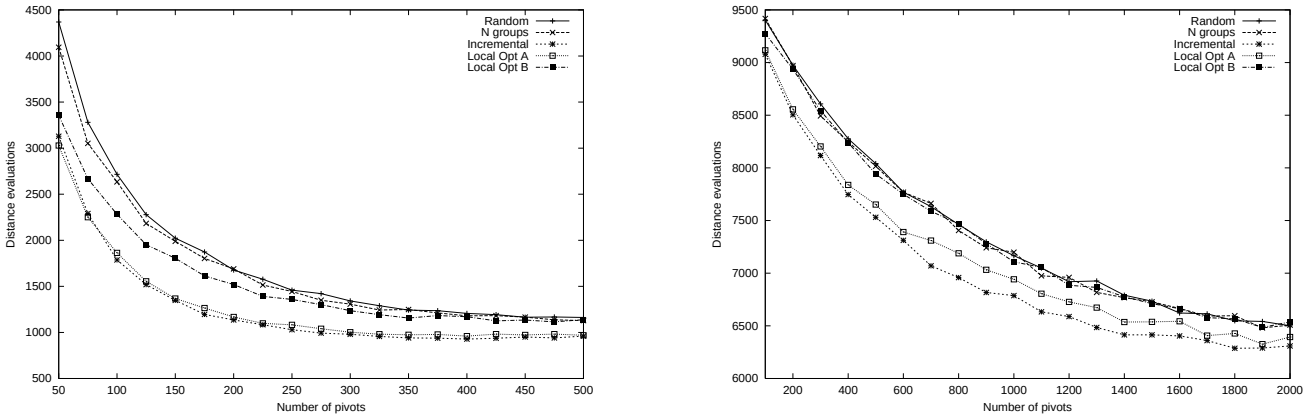

Figure 3. Comparison between selection techniques in random vector spaces of dimension 16 (left) and dimension 24 (right).

4.2. Incremental selection

A pivot p_1 is selected from a sample of N elements of the database, such that that pivot alone has the maximum μ_D value. Then, a second pivot p_2 is chosen from another sample of N elements of the database, such that $\{p_1, p_2\}$ has the maximum μ_D value, considering p_1 fixed. The third pivot p_3 is chosen from another sample of N elements, such that $\{p_1, p_2, p_3\}$ has the maximum μ_D value, considering p_1 and p_2 fixed. The process is repeated until k pivots have been chosen.

Optimization cost: If the distances $D_{\{p_1,\ldots,p_{i-1}\}}([a_r],[a'_r]), \forall r \in 1\ldots A$ are kept in an array, it is not necessary to do all the work to estimate μ_D when the i-th pivot is added. It is enough to calculate $D_{p_i}([a_r],[a'_r]), \forall r \in 1\ldots A$, because $D_{\{p_1,\ldots,p_i\}}([a_r],[a'_r]) = \max(D_{\{p_1,\ldots,p_{i-1}\}}([a_r],[a'_r]), D_{p_i}([a_r],[a'_r]))$. Therefore, only $2NA$ distance evaluations are needed to estimate μ_D when a new pivot is added. Since the process is reapeated k times, the total optimization cost is $2kAN$ distance evaluations.

4.3. Local optimum selection

A group of k pivots are chosen at random among the elements of the database. The matrix $M(r,j) = D_{p_j}([a_r],[a'_r])$, $r = 1\ldots A$, $j = 1\ldots k$ is calculated using the A pairs of elements. It follows that $D([a_r],[a'_r]) = \max_{1\le i\le k}(M(r,j))$ for every r, and this can be used to estimate μ_D. Also, it must be kept for each row of M the index of the pivot where the maximum value is, which will be denoted r_{max}, and the second maximum value, denoted r_{max2}. The *contribution* of the pivot p_j is the sum over the A rows of how much does p_j help increase the value of $D([a_r],[a'_r])$, that is $M(r,r_{max}) - M(r,r_{max2})$ if $j = r_{max}$ for that row, and 0 otherwise.

The pivot whose contribution to the value of μ_D is minimal with respect to the other pivots is marked as the *victim*, and it is replaced, when possible, by a better pivot selected from a sample of X elements of the database. The process is repeated N' times.

Optimization cost: The construction cost of the initial matrix M is $2Ak$ distance evaluations. The search cost of the victim is 0, because no extra distance evaluations are needed, all information is in M. Finding a better pivot from the X elements sample costs $2AX$ distance evaluations, and the process is repeated N' times, so the total optimization cost is $2A(k+N'X)$ distance evaluations. Considering $kN = k + N'X$, i.e. $N'X = k(N-1)$, the optimization cost is $2AkN$ distance evaluations.

Note that it is posible to exchange the values of N' and X while mantaining the optimization cost. In the experiments we use two possible value selections: $(N' = k) \wedge (X = N-1)$ (called *local optimum A*) and $(N' = N-1) \wedge (X = k)$ (called *local optimum B*). We also try with another value selection, $N' = X = \sqrt{k(N-1)}$, but the obtained result does not show any improvement on the algorithm performance.

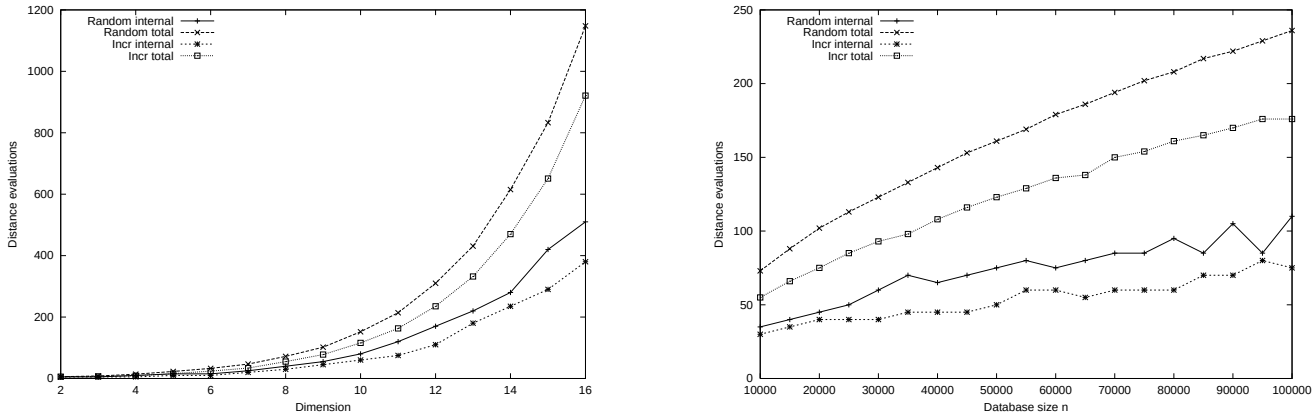

Figure 4. Comparison between random and incremental selection (using an optimal number of pivots) when varying the dimension (left) and the database size (right).

4.4. Some advantages of the incremental selection

The only way to determine the optimum number of pivots k^*, for a fixed tolerance range, is calculating an average of the total complexity of the algorithm for different values of k, where k^* is equal to the value of k which minimizes the total complexity. That is, it is worth to add pivots to the index until the total complexity does not improve.

The incremental selection technique for choosing pivots allows us to add more pivots to the index at any time without doing all the optimization work again, if the distances $D_{\{p_1,\ldots,p_k\}}([a_r],[a'_r])$, $\forall r \in 1\ldots A$ are kept. On the other hand, selection of N random groups and local optimum selection techniques must redo all the optimization work to obtain a new set of pivots, because these techniques can not take advantage of the work done previously.

For this reason, it is much easier to calculate the optimum number of pivots k^* using the incremental selection technique.

5. Experimental results

We have tested the selection techniques on a synthetic set of random points in a k-dimensional vector space treated as a metric space, that is, we have not used the fact that the space has coordinates, but treated the points as abstract objects in an unknown metric space. The advantage of this choice is that it allows us to control the exact dimensionality we are working with, which is very dificult to do in general metric spaces. The points are uniformily distributed in the unitary cube, our tests use the L_2 (Euclidean) distance, the dimension of the vector space is in the range $2\ldots24$, the database size is $n = 10,000$ (except when otherwise stated) and we perform range queries returning 0.01% of the total database size, taking an average from 1,000 queries.

About the parameters A and N of the optimization cost: Our experiments show that, given an amount of work to spend, it is better to have a high value of A and a low value of N. This indicates that it is worth to make a good estimation of μ_D, while small samples of candidate elements suffice to obtain good sets of pivots. For the experiments in this section these parameters have fixed values as follows: $A = 10,000$ and $N = 20$.

5.1. Comparison between the selection techniques

Figure 3 shows the comparison between all the selection techniques, when varying the number of pivots and keeping the dimension of the space fixed. This results show that the incremental selection technique is the one that obtains the best performance in practice, but there is no big difference with local optimum A selection, although this difference increases with larger dimensions. Local optimum B and selection of N random groups show no great improvement over random selection even in low dimensions.

Since incremental and local optimum A selection give the same efficiency, we choose the former tech-

nique as our method for choosing pivots. The reasons are those stated in Section 4.4, and that incremental selection is a much simpler technique.

5.2. Comparison between random selection and incremental selection

Figure 4 shows a comparison for internal and total complexity (see Section 2) between random and incremental selection when using the optimum number of pivots for each technique. The left plot shows a comparison when varying the dimension of the space. Since k^* is equal to the internal complexity of the algorithm, it follows that not only the optimum number of pivots is lower when using the incremental selection, but so is also the total complexity of the algorithm. The right plot shows a comparison in a vector space of dimension 8 and varying the database size. Again we obtain that the optimum number of pivots and the total complexity of the algorithm is lower when using the incremental selection.

The profit when using k^* pivots with incremental selection seems low in high dimensional spaces. However, consider that much fewer pivots (i.e. less memory) are needed to obtain the same result than with random selection. Figure 5 shows an example of this in a vector space of dimension 16. $k = 500$ is the optimum number of pivots using random selection, while incremental selection only needs 200 pivots to achieve the same total complexity, hence saving 60% of the memory used in the index.

The results obtained show that the incremental selection technique effectively produces good sets of pivots.

5.3. Properties of a good set of pivots

When studying the characteristics of the good sets of pivots, we found that good pivots have the following properties:

- Good pivots are *far away* from each other, i.e., the mean distance between pivots is higher than the mean distance between random elements of the metric space.
- Good pivots are *far away* from the rest of the elements of the metric space.

The elements that satisfy these properties are called *outliers*. It is clear that pivots *must be far away from each other*, because two very close pivots give almost the same information for discarding elements. This is in accordance with previous observations [9, 12, 4].

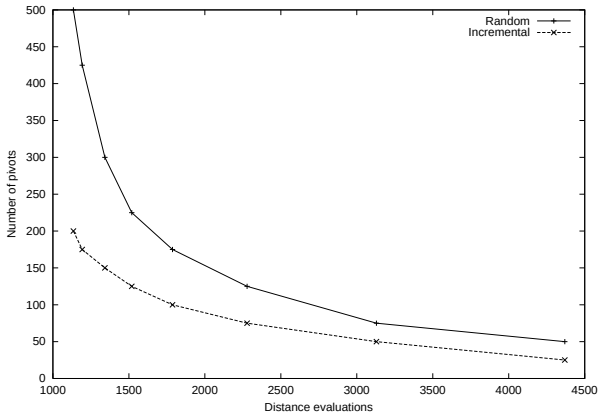

Figure 5. Number of pivots needed to answer range queries using random and incremental selection with the same total complexity.

Then, it can be assumed that good pivots are outliers, so a new selection technique could be as follows: use the same incremental selection method with the new criterion of selecting elements which maximize the sum of the distances between the pivots previously chosen, selecting the first pivot at random. This technique will be called *outliers selection*. It carries out $(i-1)N$ distance evaluations when the i-th pivot is added, where N is the size of the sample of elements from where a new pivot is selected. Hence, the optimization cost of this selection technique is $\frac{k(k-1)}{2}N$.

It is important to note that outliers selection *do not use the efficiency criterion described in Section 3*, because this alternative selection technique maximizes the mean distance in the original space and the efficiency criterion maximizes the mean of distance D. These criteria do not always go together.

5.4. Comparison between incremental selection and outliers selection

Figure 6 shows the result obtained when comparing incremental and outliers selection techniques in random vector spaces. The figures show that the outliers selection has better performance that the incremental selection. This result can lead to think that outliers selection is the best pivot selection technique, but in the next section we will see that this assumption is not true for general metric spaces.

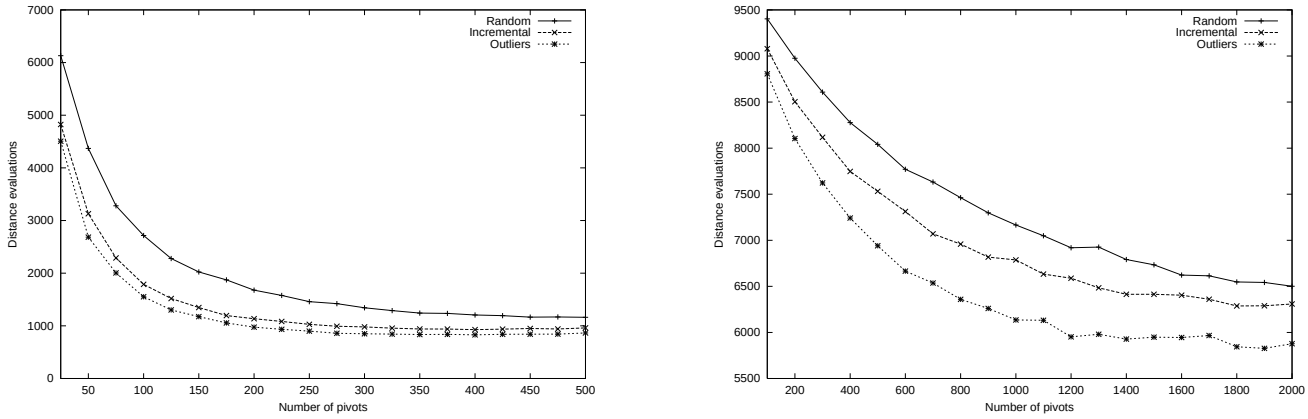

Figure 6. Comparison between incremental and outliers selection techniques in random vector spaces of dimension 16 (left) and dimension 24 (right).

5.5. Real-world examples

Now we present three examples of the use of the incremental selection and the outliers selection, where the elements of the metric space are not uniformely distributed.

Figure 7 (left) shows the results of the experiment when the elements of the database are a set of 40,700 images from NASA archives [1]. Those images were transformed into 20-dimensional vectors, and the 10% of the database was defined as the query set. We used a tolerance range which returns on average 0.01% of the elements of the database per query. The figure shows that for more than 25 pivots the outliers selection technique has worse performance that the random selection, while incremental selections always performs better. This result is in contrast with those obtained on uniformly distributed vector spaces.

Figure 7 (right) shows the results of the experiment when the elements of the database are points in a 30-dimensional vector space, where the elements are not uniformly distributed but have a Gaussian distribution, that is, the elements form clusters. The result shows that both incremental and outliers selection improve the performance of the algorithm in comparison with the random selection, but incremental selection performs better for few pivots.

Figure 8 shows the results of the experiment over a string space, that is, the elements of the database were strings taken from a Spanish dictionary of about 80,000 terms, and a 10% of the database was used as the query set. The distance function used was the *edit distance* (the minimum number of character insertions, deletions and substitutions to make two strings equal), and the tolerance range was $r = 2$, which retrieves an average of 0.02% of the database size per query. In this case the incremental selection improves the performance of the algorithm with respect to the random selection, while the outliers selection obtained worse performance than with random selection.

6. Conclusions

We have defined an efficiency criterion to compare two sets of pivots, and have shown experimentally that this criterion consistently selects good sets of pivots in a variety of synthetic and real-world metric spaces, reducing the total complexity of pivot-based proximity searching when answering range queries. We presented three different pivot selection techniques, which use the efficiency criterion defined, and showed that the so-called *incremental selection technique* is the best selection method in practice. We have found that good pivots have the property of being outliers, but outliers are not necesarily good pivots. It is interesting to note that outliers sets have good performance in uniformly distributed vector spaces, but have bad performance in general metric spaces, even worse than random selection in some cases. This result leads to questioning if it is valid to test pivot selection techniques in uniformly distributed vector spaces.

Future work involves testing some new heuristics for pivots selection (e.g. select pivots from a set of outliers previously chosen from the database), and testing alternative efficiency estimators (e.g. select pivots that maximize the minimum D distance of the histogram),

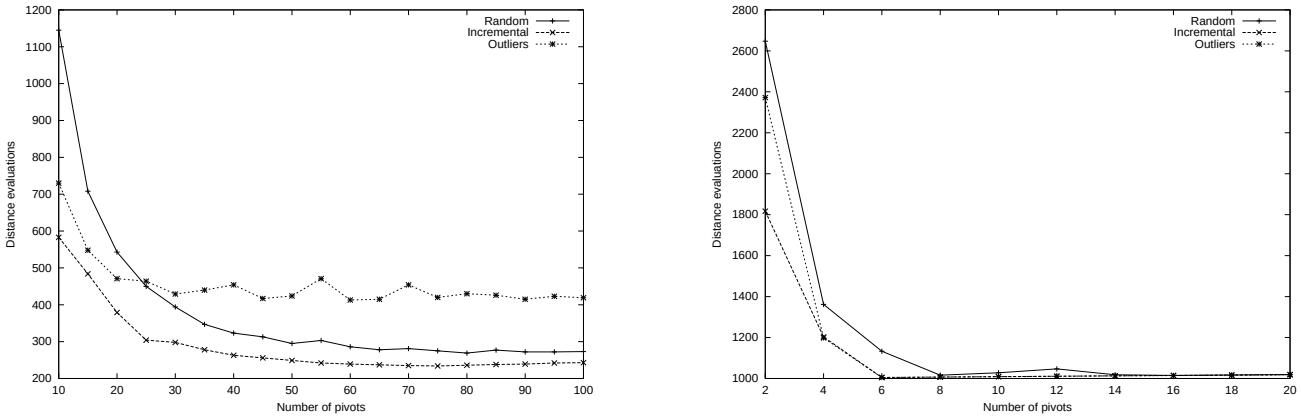

Figure 7. Experiments with NASA images database (left) and a vectorial space with Gaussian distribution (right).

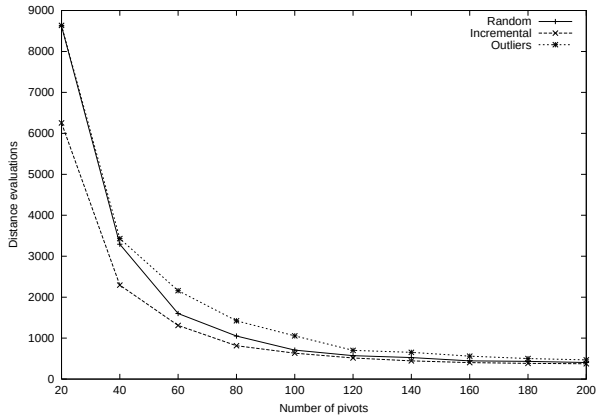

Figure 8. Experiment with a string metric space

always with the aim of maximizing the probability of discarding elements (Eq. 2).

References

[1] The Sixth DIMACS Implementation Challenge: Available Software, `http://www.dimacs.rutgers.edu/-Challenges/Sixth/software.html`.

[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] T. Bozkaya and M. Ozsoyoglu. Distance-based indexing for high-dimensional metric spaces. In *Proc. ACM SIGMOD International Conference on Management of Data*, pages 357–368, 1997. Sigmod Record 26(2).

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

[5] W. Burkhard and R. Keller. Some approaches to best-match file searching. *Comm. of the ACM*, 16(4):230–236, 1973.

[6] E. Chávez, J. Marroquín, and R. Baeza-Yates. Spaghettis: an array based algorithm for similarity queries in metric spaces. In *Proc. String Processing and Information Retrieval (SPIRE'99)*, pages 38–46. IEEE CS Press, 1999.

[7] E. Chávez, J. Marroquín, 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. To appear.

[8] E. Chávez, G. Navarro, R. Baeza-Yates, and J. Marroquín. Proximity searching in metric spaces. *ACM Computing Surveys*, 2001. To appear.

[9] A. Faragó, T. Linder, and G. Lugosi. Fast nearest-neighbor search in dissimilarity spaces. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 15(9):957–962, 1993.

[10] L. Micó, J. Oncina, and E. Vidal. A new version of the nearest-neighbor approximating and eliminating search (AESA) with linear preprocessing-time and memory requirements. *Pattern Recognition Letters*, 15:9–17, 1994.

[11] E. Vidal. An algorithm for finding nearest neighbors in (approximately) constant average time. *Pattern Recognition Letters*, 4:145–157, 1986.

[12] P. Yianilos. Data structures and algorithms for nearest neighbor search in general metric spaces. In *Proc. 4th ACM-SIAM Symposium on Discrete Algorithms (SODA'93)*, pages 311–321, 1993.

[13] P. Yianilos. Excluded middle vantage point forests for nearest neighbor search. In *DIMACS Implementation Challenge, ALENEX'99*, Baltimore, MD, 1999.