Towards Measuring the Searching Complexity of Metric Spaces ⋆

Edgar Chávez[1] and Gonzalo Navarro[2]

[1] Esc. Cs. Físico-Matemáticas
Universidad Michoacana elchavez@fismat.umich.mx
[2] Depto. de Ciencias de la Computación
Universidad de Chile gnavarro@dcc.uchile.cl

Abstract. In this paper we introduce a new measure of the intrinsic searching complexity of a general metric space. This measure reflects the expected behavior of the search algorithms on the metric space, yet it is easy to estimate and independent of the search algorithm. We prove average case lower bounds, in terms of this complexity measure, for a large class of proximity search algorithms. This gives some new insight on the intrinsic difficulty of the search problem in metric spaces.

1 Introduction

Proximity searching in metric spaces consists in finding either the (k) nearest neighbors or the set of points within a fixed distance (r) to a query element. Due to its relevance in a vast number of fields, from multimedia retrieval to machine learning, classification and Web searching, a lot of practical algorithms have been proposed. Those algorithms are effective in some metric spaces (e.g. vector spaces of intrinsic low dimension), but perform poorly in some other spaces (e.g. DNA subsequences, and text collections with the edit distance).

It has been suggested that the histogram of distances between database elements is an important measure of how difficult it is to search for proximity queries in a particular instance of metric space. Under this point of view, the easy instances would have a flat histogram, while the hard spaces have a sharp histogram [1].

For vector spaces, it has been suspected for a long time that no algorithm could break the so called "curse of dimensionality". Despite that the problem can be trivially solved in $O(1)$ time using space exponential on the dimension, the research focuses on the more realistic case of polynomial space. Under this case, one version of the curse of dimensionality has been recently proved for nearest neighbor searching on the Hamming cube $(\{0,1\}^d)$ in the worst case sense. Chakrabarti et al. [2] obtain $\Omega(\log\log d/\log\log\log d)$ for approximate algorithms, while Borodin et al. [3] give a lower bound of $\Omega(\log d)$ for exact algorithms, even allowing randomization. On the other hand, there exist $O(\log\log d)$ approximate randomized algorithms [4]. There exist also previous $\Omega(n^\alpha)$ worst case lower bounds for orthogonal range searching on a set of n real d-dimensional points by Melhorn [5], but this does not consider the dimension d.

⋆ This project has been partially supported by CYTED VII.13 AMYRI Project, Fondecyt Grant 1-000929 and CONACyT Project R-28923A.

For general metric spaces, on the other hand, there are no known lower bounds, neither in the average nor in the worst case sense. In most cases even the analyses of particular algorithms seem so difficult that the authors validate their complexity claims just by experiments [6,7] A few authors attempt to formally analyze their algorithms [8–11,1], but they need to make simplifying assumptions that have to be experimentally validated anyway.

In [12] we presented a framework able to unify the existing approaches under a unique theoretical model. This paper is aimed at discovering a measure of the intrinsic search difficulty (which is independent of the search algorithm) of a metric space. To this end we analyze the complexity of a large class of proximity search algorithms and prove lower bounds in terms of this measure of intrinsic search difficulty.

2 Basic Concepts

The set $\mathbb{X}$ will denote the universe of *valid* objects. A finite subset of it, $\mathbb{U}$, of size $n = |\mathbb{U}|$, is the set of objects, or database, where we search. A distance function $d : \mathbb{X} \times \mathbb{X} \longrightarrow \mathbb{R}$, satisfying the usual properties of triangle inequality, reflexivity, symmetry and positiveness, will measure the closeness between points. The metric space is the pair $(\mathbb{X}, d)$.

There are basically two types of queries of interest in metric spaces:

Range query: Retrieve all elements which are within distance r to q.
This is, retrieve $(q,r)_d \;=\; \{u \in \mathbb{U} \;/\; d(q,u) \leq r\}$.

(k) Nearest neighbor query: Retrieve the k closest elements to q in $\mathbb{U}$.
This is, retrieve $nn_d(q,k) \subseteq \mathbb{U}$ such that $|nn_d(q,k)| = k$ and $\forall u \in nn_d(q,k), v \in \mathbb{U} - nn_d(q,k),\; d(q,u) \leq d(q,v)$.

In this work we concentrate on range queries for simplicity. Many of the results, however, can be extended to nearest neighbor searching as well, since the corresponding algorithms are normally built over those for range queries [13].

Since the operation of leading complexity is computing distances, we will measure the complexity of the searching algorithms in terms of the number of distances computed.

3 Proximity Search Algorithms

Different data structures have been proposed to filter out elements based on the triangular inequality (see [13] for a complete survey). We divide the exposition according to the two main techniques used.

Pivot-based algorithms are built on a single general idea: select some elements from $\mathbb{U}$ (called *pivots*), and identify all the other elements with their distances to (some of) the pivots. The methods differ in how they select the pivots, how much information they store about the distances among elements and pivots, etc. The simplest variant works as follows: k pivots are selected and each object v is mapped to k coordinates which are its distances to the pivots:

$\Phi(v) = (d(v,p_1), \ldots, d(v,p_k))$. Later, the query q is also mapped to $\Phi(q)$ and if it differs from an object in more than r along some coordinate then the element is filtered out by the triangle inequality. That is, if for some pivot p_i and some element v of the set it holds $|d(q,p_i) - d(v,p_i)| > r$, then we know that $d(q,v) > r$ without need to evaluate $d(v,q)$. The elements that cannot be filtered out using this rule are directly compared.

An interesting feature of most of these algorithms is that they can reduce the number of distance evaluations by increasing the number of pivots. Define $D_k(x,y) = L_\infty(\Phi(x), \Phi(y)) = \max_{1 \leq j \leq k} |d(x,p_j) - d(y,p_j)|$. Using the pivots $p_1, ..., p_k$ is equivalent to discarding elements u such that $D_k(q,u) > r$. As more pivots are added we need to perform more distance evaluations (exactly k) to compute $D_k(q,*)$ (these are called *internal* evaluations), but on the other hand $D_k(q,*)$ increases its value and hence it has a higher chance of filtering out more elements (those comparisons against elements that cannot be filtered our are called *external*). It follows that there exists an optimum k. This optimum, however, cannot be normally reached because it is too high in terms of space requirements: kn distances have to be precomputed and stored in order to use k pivots. Hence, in general these methods use as many pivots as they can, and they are normally well below their optimum.

Clustering algorithms try to divide the space in zones as compact as possible. They select a set of *centers*, which are elements from $\mathbb{U}$, and divide the space so that each center has its zone of influence. Each zone is normally divided recursively. The algorithms differ in how the centers are selected, how the zones are delimited, etc.

4 The Intrinsic Search Difficulty of a Metric Space

For the specific goal of predicting the behavior of an indexing data structure, the authors of [14] propose a measure called the *distance exponent*, based on an empirical power law observed in many data sets. They note that the average number of sites within distance r to a database element is proportional to r^α. Extrapolating this feature, they derive formulas for selectivity estimation of range queries, and relate α to the intrinsic dimension of the metric data set. Other articles relevant to the relationship between intrinsic dimension and search cost are [15, 16, 1].

Despite all the efforts, no definition of "dimension" in metric spaces has provided a lower bound for the search complexity, as they have been obtained for metric spaces. Our goal is to define a measure of the intrinsic search difficulty which, albeit not necessarily related to a concept of dimension, permits us deriving those lower bounds.

Many authors [6, 17, 1] have proposed to use the histogram of distances to characterize the difficulty of searching in an arbitrary metric space, but no quantitative definition has been attempted. We present now a quantitative measure in this line and study its suitability.

Let us start with a well-known example. Consider a distance such that $d(x,x) = 0$ and $d(x,y) = 1$ for all $x \neq y$. Under this distance (in fact an equality test), we do not obtain any information from a comparison except that the element considered is or is not our query. It is clear that it is not possible to avoid a sequential search in this case, no matter how smart is our indexing technique.

Let us consider the *histogram* of distances between points in the metric space $\mathbb{X}$. This can be approximated by using the dictionary $\mathbb{U}$ as a random sample of $\mathbb{X}$. The idea is that, as the space is more difficult to search, the mean μ of the histogram grows and/or its variance σ^2 is reduced. Our previous example is an extreme case.

Figure 1 gives an intuitive explanation of why the search problem is harder when the histogram is concentrated. If we consider a random query q and an indexing scheme based on random pivots, then the possible distances between q and a pivot p are distributed according to the histogram of the figure. The elimination rule says that we can discard any point u such that $d(p,u) \notin [d(p,q) - r, d(p,q) + r]$. The grayed areas in the figure show the points that we *cannot* discard. As the histogram is more and more concentrated around its mean, less and less points can be discarded using the information given by $d(p,q)$. Moreover, in many cases (e.g. to retrieve a fixed fraction of the database) the search radius r must grow as the mean distance μ grows, which makes the problem even harder.

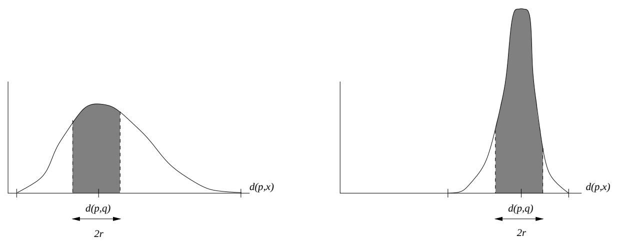

Fig. 1. An easy (left) and hard (right) histogram of distances, showing that on hard histograms virtually all the elements become candidates for exhaustive evaluation.

Note that in general the histogram can have more than one peak, especially with clustered data. However, after using just one pivot we remove from consideration all the peaks non relevant to the query. Hence, the difficult part of the search starts when the histogram of (remaining) elements has just one peak.

This phenomenon is independent of the nature of the metric space (vectorial or not, in particular) and gives us a way to quantify how hard is to search on an arbitrary metric space.

Definition. *The* intrinsic search difficulty *of a metric space is defined as* $\rho = \frac{\mu^2}{2\sigma^2}$, *where* μ *and* σ^2 *are the mean and variance of its histogram of distances.*

The technical convenience of the exact definition is made clear shortly. Observe that the intrinsic search difficulty grows with the mean and with the inverse of the variance of the histogram. Moreover, measuring this intrinsic difficulty on an arbitrary and unknown metric space can be accomplished by simple statistical means via a reasonable number of distance evaluations among random points of the set. This is much simpler and cheaper than all previous approaches, in particular those based on a definition of dimension.

On multimodal distributions we can consider one peak of the histogram at a time and obtain a search difficulty for queries lying in each corresponding cluster, or consider the whole histogram and obtain a general lower bound (lower than the separate bounds).

Let us check our definition on vector spaces. As shown in [18], a uniformly distributed ℓ-dimensional vector space under the L_s distance has mean $\Theta(\ell^{1/s})$ and standard deviation $\Theta(\ell^{1/s-1/2})$. Therefore its intrinsic difficulty is $\Theta(\ell)$ (although the constant is not necessarily 1). So the intuitive concept of dimensionality in vector spaces matches our general concept of intrinsic search difficulty. After all, there is a relation to dimension, where it can be defined.

We have generated random uniformly distributed points in the real ℓ-dimensional vector space $[0,1)^\ell$ for ℓ between 2 and 20, using three different distances L_1, L_2 and L_∞. We have selected one million pairs of points for each combination of dimension ℓ and distance L_s and have computed the intrinsic difficulty $\mu^2/(2\sigma^2)$ of the resulting histogram of distances. The intrinsic difficulty grows linearly with the representational dimension when the points are chosen at random. We have included a table with the least squares estimations, which shows that a vector space of dimension ℓ has intrinsic difficulty from $1.00 \times \ell$ to $1.43 \times \ell$, depending on the L_s distance used (any L_s with $s \geq 3$ generates a line between those of L_2 and L_∞).

Distance	Intrinsic Difficulty	Percent Error
L_1	$-0.314 + 1.003\ell$	0.34%
L_2	$-1.182 + 1.346\ell$	1.66%
L_∞	$-1.631 + 1.425\ell$	3.27%

5 Lower Bounds in Terms of the Intrinsic Difficulty

5.1 A Lower Bound for Pivoting Algorithms

Our main result in this section is a lower bound on the range search cost with a given search radius r using a pivoting scheme that chooses k global pivots at random. As we show next, the complexity of the search is related to r and the intrinsic difficulty ρ.

We are considering independent identically distributed (i.i.d.) random variables for the distribution of distances between points. Although not accurate, this simplification is optimistic and hence can be used to lower bound the performance of the indexing algorithms. We come back to this shortly.

Let $(q,r)_d$ be a range query over a metric space indexed by means of k random pivots, and let u be an element of $\mathbb{U}$. The probability that u cannot be excluded from direct verification after considering the k pivots is exactly

$$Pr(|d(q,p_1) - d(u,p_1)| \leq r, \ldots, |d(q,p_k) - d(u,p_k)| \leq r) \qquad (1)$$

Since all the pivots are assumed to be random and their distance distributions i.i.d. random variables, this expression is the product of probabilities

$$Pr(|d(q,p_1) - d(u,p_1)| \leq r) \ \times \ \ldots \ \times \ Pr(|d(q,p_k) - d(u,p_k)| \leq r) \qquad (2)$$

which for the same reason can be simplified to

$$Pr(\text{not discarding } u) \ = \ Pr(|d(q,p) - d(u,p)| \leq r)^k \qquad (3)$$

for a random pivot p.

If X and Y are two i.i.d. random variables with mean μ and variance σ^2, then the mean of $X - Y$ is 0 and its variance is $2\sigma^2$. Using Chebyschev's inequality[1] we have that $Pr(|X-Y| > \varepsilon) < 2\sigma^2/\varepsilon^2$. Therefore,

$$Pr(|d(q,p) - d(u,p)| \leq r) \ \geq \ 1 - \frac{2\sigma^2}{r^2}$$

where σ^2 is precisely the variance of the distance distribution in our metric space. The argument that follows is valid for $2\sigma^2/r^2 < 1$, or $r > \sqrt{2}\sigma$ (large enough radii), otherwise the lower bound is zero. Then, we have

$$Pr(\text{not discarding } u) \ \geq \ \left(1 - \frac{2\sigma^2}{r^2}\right)^k$$

We have now that the total search cost is the number of internal distance evaluations (k) plus the external evaluations (those to check the remaining candidates), whose number is on average $n \times Pr(\text{not discarding } u)$. Therefore

$$Cost \ \geq \ k \ + \ n \left(1 - \frac{2\sigma^2}{r^2}\right)^k$$

is a *lower bound* to the average search cost by using pivots. Optimizing we obtain that the best k is

$$k^* \ = \ \frac{\ln n \ + \ \ln\ln(1/t)}{\ln(1/t)}$$

where $t = 1 - 2\sigma^2/r^2$. Using this optimal k^*, we obtain an absolute (i.e. independent on k) lower bound for the average cost of any random pivot-based algorithm:

$$Cost \ \geq \ \frac{\ln n \ + \ \ln\ln(1/t) \ + \ 1}{\ln(1/t)} \ \geq \ \frac{\ln n}{\ln(1/t)} \ \geq \ \frac{r^2}{2\sigma^2} \ln n$$

[1] For an arbitrary distribution Z with mean μ_z and variance σ_z^2, $Pr(|Z - \mu_z| > \varepsilon) < \sigma_z^2/\varepsilon^2$.

which shows that the cost depends strongly on σ/r. As r increases t tends to 1 and reaching the optimum requires more and more pivots, yet it gets more and more costly.

Another way to represent this result is to assume that we are interested in retrieving at most a fixed fraction f of the elements, in which case r can be written as $r = \mu - \sigma/\sqrt{f}$ by Chebyschev's inequality. In this case the lower bound becomes

$$\frac{r^2}{2\sigma^2} \ln n \quad = \quad \frac{(\mu - \sigma/\sqrt{f})^2}{2\sigma^2} \ln n \quad = \quad \left(\sqrt{\rho} - \frac{1}{\sqrt{2f}}\right)^2 \ln n$$

which is valid for $f \geq 1/(2\rho)$. We have just proved

Theorem 1. *Any pivot based algorithm using random pivots has a lower bound $(\sqrt{\rho} - 1/\sqrt{2f})^2 \ln n$ in the average number of distance evaluations performed for a random range query retrieving at most a fraction f of the set, where ρ is the intrinsic search difficulty of the metric space.*

This result matches that of [19,20] on FHQTs, about obtaining $\Theta(\log n)$ search time using $\Theta(\log n)$ pivots, but here we are more interested in the "constant" term, which depends on the metric space itself and not on the database size n.

The theorem shows clearly that the parameters governing the performance of range searching algorithms are ρ and f. As ρ grows and f stays fixed, this tends to $\rho \ln n$.

We have considered i.i.d. random variables for each pivot and the query. This is a reasonable approximation, as we do not expect much difference between the "view points" from the general distribution of distances to the individual distributions, a subject discussed in depth in [1]. The expression given in Eq. (3) cannot be obtained without this simplification.

A stronger assumption comes from considering all the variables as independent. This is an optimistic consideration equivalent to assuming that in order to discard each element u of the set we take k new pivots at random. The real algorithm fixes k random pivots and uses them to try to discard all the elements u of the set. The latter alternative can suffer from dependencies from a point u to another, which cannot happen in the former case (for example, if u is close to a pivot p and u' is close to u then the distance from u' to p carries less information). Since the assumption is optimistic, using it to reduce the joint distribution in Eq. (1) to the expression given in Eq. (2) keeps the lower bound valid.

5.2 The Target Space

It is interesting to study which is the behavior of the mapped space after selecting k pivots. Let us start with the mean of the distance D_k in the target space. As already seen, $D_k(u,v)$ is the maximum of k random variables $|d(u,p_i) - d(v,p_i)|$, and therefore

$$Pr(D_k(u,v) \leq \varepsilon) \quad = \quad Pr(|d(u,p) - d(v,p)| \leq \varepsilon)^k \quad \geq \quad \left(1 - \frac{2\sigma^2}{\varepsilon^2}\right)^k$$

where the last inequality comes from applying Chebyschev as in the previous section. Calling Z the random variable associated to D_k, we have that its cumulative probability function is

$$F_z(x) \quad = \quad Pr(D_k(u,v) \leq x) \quad \geq \quad \left(1 - \frac{2\sigma^2}{x^2}\right)^k$$

if $x \geq \sqrt{2}\sigma$. $F_z(x)$ can be optimistically assumed to be zero otherwise ("optimistically" means that we will obtain an upper bound on $E(Z)$). To obtain the density function $f_z(x)$ we differentiate the cumulative distribution and get

$$f_z(x) \quad = \quad k(1 - 2\sigma^2/x^2)^{k-1} 4\sigma^2/x^3$$

if $x \geq \sqrt{2}\sigma$ and zero otherwise. Now to obtain the mean we compute

$$E(Z) \quad \leq \quad \int_{\sqrt{2}\sigma}^{\infty} x f_z(x)\, dx \quad = \quad 4k\sigma^2 \int_{\sqrt{2}\sigma}^{\infty} (1 - 2\sigma^2/x^2)^{k-1}/x^2\, dx$$

Now doing the change $y = 2\sigma^2/x^2$ (and hence $dy = -4\sigma^2/x^3\, dx$) we have

$$E(Z) \quad \leq \quad \sqrt{2}k\sigma \int_0^1 (1-y)^{k-1} y^{-1/2}\, dy \quad = \quad \sqrt{2}k\sigma \binom{k+\frac{1}{2}}{\frac{1}{2}}$$

This is the exact solution (recall that $\binom{k+1/2}{1/2} = (k+1/2)(k-1/2)(k-3/2)\cdots(5/2)(3/2)$). For large k this converges to

$$E(Z) \quad \leq \quad \sigma\sqrt{2\pi k}$$

This means that the mean value of D_k is independent on the mean μ of the original metric space. Rather, it is proportional to the standard deviation (this is reasonable because it is a maximum of differences between distances). On the other hand, this mean grows with the square root of the number of pivots. Recall that, since this comes from using Chebyschev's inequality, the result is just an upper bound on the mean of D_k.

It is clear that D_k, as a lower bound for d, should be as close to d as possible in order to filter out most of the irrelevant elements. Therefore, we could like that both means be equal. Solving $\mu = \sigma\sqrt{2\pi k}$ yields $k = \rho/\pi$, which means that we must have at least a number of pivots proportional to the intrinsic difficulty of the space. This is an optimistic bound and in practice k has to be much larger (as shown in Theorem 1, the optimum is indeed larger).

We have tried to obtain an upper bound on the variance of D_k, but the integral $\int x^2 f_z(x)\, dx$ does not converge. This does not mean that D_k does not have variance, because ours is just an upper bound.

This also shows that we could search in the mapped space with radius $r' = r\sqrt{2}k\sigma\binom{k+1/2}{1/2}/\mu = rk\binom{k+1/2}{1/2}/\sqrt{\rho} \approx \sqrt{\pi k}r/\sqrt{\rho}$ and still get most of the results. This opens the door to probabilistic algorithms which, with the penalty of a small error probability, are much faster than the exact versions [21].

5.3 A Lower Bound for Clustering Algorithms

With a similar procedure we can prove a lower bound for clustering algorithms. For space limitations we do not give a proof of the theorem. The technical report contains all the details [22].

Theorem 2. *Any clustering based algorithm using random centers has a lower bound* $\mathcal{C}_I(2(\sqrt{\rho} - 1/\sqrt{2f})^2)$ *in the average number of distance evaluations performed for a random range query retrieving a fraction of at most* f *of the database, where* ρ *is the intrinsic dimension of the space and* $\mathcal{C}_I()$ *is the internal complexity to find the relevant classes, satisfying* $\Omega(\log m) = \mathcal{C}_I(m) = O(m)$*.*

This result is weaker than Theorem 1 because of our inability to give a good lower bound on $\mathcal{C}_I$, so we cannot ensure more than a logarithmic increase with respect to ρ. However, even assuming $\mathcal{C}_I(m) = \Theta(m)$ (i.e. exhaustive search of the classes), when the theorem becomes very similar to Theorem 1, there is an important reason that explains why the clustering based algorithms can in practice be better than pivot based ones. We *can* in this case achieve the optimal number of centers m^*, which is impossible in practice for pivot-based algorithms. The reason is that it is much more economical to represent the Voronoi partition using m centers than the pivot partition using k pivots.

6 Conclusions and Future Work

We have proposed a new quantitative measure of the intrinsic search difficulty of a general metric space. It is simple and, unlike previous attempts, it permits deriving lower bounds on the search cost of large classes of proximity search algorithms.

Future work involves tuning this measure so as to increase its predictive power (currently it serves only as a lower bound), therefore giving a practical and extremely useful tool to predict the difficulty of searching on an unknown metric space.

References

1. Ciaccia, P., Patella, M., Zezula, P.: A cost model for similarity queries in metric spaces. In: Proc. 17th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS'98). (1998)
2. Chakrabarti, A., Chazelle, B., Gum, B., Lyov, A.: A lower bound on the complexity of approximate nearest-neighbor searching on the hamming cube. In: 31st Annual ACM Symp. Theory of Computing. (1999) 305–311
3. Borodin, A., Ostrovsky, R., Rabani, Y.: Lower bounds for high dimensional nearest neighbor search and related problems. In: 31st Annual ACM Symp. Theory of Computing. (1999)
4. Indyk, P., Motwani, R.: Approximate nearest neighbors: towards removing the curse of dimensionality. In: 30th Annual ACM Symp. Theory of Computing. (1998) 604–613

5. Melhorn, K.: Data Structures and Algorithms. Volume III - Multidimensional Searching and Computational Geometry. Springer (1984)
6. Brin, S.: Near neighbor search in large metric spaces. In: Proc. 21st Conference on Very Large Databases (VLDB'95). (1995) 574–584
7. Vidal, E.: An algorithm for finding nearest neighbors in (approximately) constant average time. Pattern Recognition Letters **4** (1986) 145–157
8. Clarkson, K.: Nearest neighbor queries in metric spaces. Discrete Computational Geometry **22** (1999) 63–93
9. Yianilos, P.: Data structures and algorithms for nearest neighbor search in general metric spaces. In: Proc. 4th ACM-SIAM Symposium on Discrete Algorithms (SODA'93). (1993) 311–321
10. Baeza-Yates, R., Cunto, W., Manber, U., Wu, S.: Proximity matching using fixed-queries trees. In: Proc. 5th Combinatorial Pattern Matching (CPM'94). LNCS 807 (1994) 198–212
11. Navarro, G.: Searching in metric spaces by spatial approximation. In: Proc. String Processing and Information Retrieval (SPIRE'99), IEEE CS Press (1999) 141–148
12. Chávez, E., Navarro, G., Baeza-Yates, R., Marroquín, J.: A unified model for similarity searching. In Ahuactzin, J., ed.: Actas del Encuentro Nacional de Computación Mexicano, Pachuca, Hidalgo, México, Sociedad Mexicana de Ciencias de la Computación (1999) CD-ROM edition.
13. Chávez, E., Navarro, G., Baeza-Yates, R., Marroquín, J.: Searching in metric spaces. Technical Report TR/DCC-99-3, Dept. of Computer Science, Univ. of Chile (1999) To appear in *ACM Computing Surveys.*
14. Traina, C., Traina, A., Faloutsos, C.: Distance exponent: a new concept for selectivity in metric trees. Technical Report CMU-CS-99-110, Carnegie Mellon Univ. (1999)
15. Beyer, K., Goldstein, J., Ramakrishnan, R., Shaft, U.: When is nearest neighbor meaninful? In: Proc. 7th International Conference on Database Theory (ICDT'99). LNCS 1540 (1999) 217–235
16. Weber, R., Schek, H.J., Blott, S.: A quantitative analysis and performance study for similarity search methods in high-dimensional spaces. In: Proc. 24th International Conference on Very Large Databases (VLDB'98). (1998) 194–205
17. Chávez, E., Marroquín, J.: Proximity queries in metric spaces. In Baeza-Yates, R., ed.: Proc. 4th South American Workshop on String Processing (WSP'97), Carleton University Press (1997) 21–36
18. Yianilos, P.: Excluded middle vantage point forests for nearest neighbor search. In: DIMACS Implementation Challenge, ALENEX'99, Baltimore, MD (1999)
19. Baeza-Yates, R.: Searching: an algorithmic tour. In Kent, A., Williams, J., eds.: Encyclopedia of Computer Science and Technology. Volume 37. Marcel Dekker Inc. (1997) 331–359
20. Baeza-Yates, R., Navarro, G.: Fast approximate string matching in a dictionary. In: Proc. 5th South American Symposium on String Processing and Information Retrieval (SPIRE'98), IEEE CS Press (1998) 14–22
21. Chávez, E., Navarro, G.: A probabilistic spell for the curse of dimensionality. In: Proc. 3rd Workshop on Algorithm Engineering and Experiments (ALENEX'01). LNCS (2001) To appear.
22. Chávez, E., Navarro, G.: Towards measuring the searching complexity of general metric spaces. Technical report, Universidad Michocana, México (2001) http://www.fismat.umich.mx/ elchavez/model.ps.gz.