The Many Facets of Approximate Similarity Search

Marco Patella and Paolo Ciaccia
DEIS, University of Bologna - Italy
{marco.patella,paolo.ciaccia}@unibo.it

Abstract

In this article, we review the major paradigms for approximate similarity queries and propose a classification schema that easily allows existing approaches to be compared along several independent coordinates. Then, we discuss the impact that scheduling of index nodes can have on performance and show that, unlike exact similarity queries, no provable optimal scheduling strategy exists for approximate queries. On the positive side, we show that optimal-on-the-average *schedules are well-defined. We complete by critically reviewing methods for evaluating the quality of approximate results.*

1. Introduction

Similarity queries are a search paradigm which is profitably used in a variety of modern applications. In its essence, the problem is to find objects which are similar, up to a given degree, to a given query object. In order to assess the similarity between pair of objects, usually a notion of distance is used, being understood that low values of distance correspond to high degrees of similarity. More formally, we are faced with the following problem: Given a metric space $\mathcal{M} = (\Omega, d)$, where Ω is a domain (the *object space*) and $d : \Omega \times \Omega \rightarrow \Re_0^+$ is a non-negative and symmetric binary function that also satisfies the triangle inequality, and a data set of objects $X \subseteq \Omega$, retrieve the object(s) in X which are closest according to d to a user-specified query object $q \in \Omega$. Examples of metric spaces include the D-dimensional vector space $\Re^D$ equipped with the Euclidean distance L_2 or the set Σ^* of (finite length) strings obtained from an alphabet of symbols Σ equipped with the edit distance d_{edit} (i.e., the minimum number of symbols that have to be inserted, deleted or substituted in order to transform a string into another). Typical similarity queries include *range* queries (where all the objects in X whose distance to q does not exceed a user-specified threshold r are requested) and k-nearest neighbors (k-NN) queries (where the k objects in X which are closest to q are requested). Since k-NN queries represent the most used type of similarity queries (because the user can control the query selectivity, i.e., the cardinality of the result set), in the following we will concentrate on this kind of queries.

Several access structures have been proposed to speed up the resolution of similarity queries: They can be broadly classified (depending on their field of applicability) as multi-dimensional (or spatial) and metric access methods (the former only apply when the feature space is a vector space). Recent studies, however, have pointed out the fact that using such access structures is sometimes not very efficient (e.g., when the feature space is a high-dimensional vector space [31, 19]): In such cases, the most efficient way to exactly solve similarity queries is to sequentially scan the entire data set, comparing each object against the query object q. Obviously, such solution is not viable for very large data sets.

To speed-up the search it is common to offer to the user a quality/time trade-off: If the user is willing to save search time, she has to accept a degradation in the quality of the result, i.e., an error with respect to the exact case. Approximate similarity search, therefore, has the goal to reduce search times for similarity queries by (possibly but not necessarily) introducing an error in the result.

In this work we review existing approximate similarity search techniques, proposing a classification schema (Section 2) able to characterize them according to different aspects. The goal is to present an unified view over the different approaches proposed in literature (Section 3). We then discuss the important problem of scheduling (Section 4), presenting original results on optimality of schedules. Finally, we review methods commonly used to evaluate the quality of approximate results (Section 5) and conclude.

1.1. Why To Approximate?

Approximate similarity search has the goal of reducing the cost of similarity queries by relaxing the correctness constraint, i.e., the approximate result might contain objects which are not among the k NNs of the query object q. The main rationale for providing the user with approximate techniques is (at least) threefold:

- First of all, a gap exists between the user-perceived similarity and the one actually implemented via the distance function. The "exact" result of a query, in many cases, might actually be deemed incorrect by the user, which would rather obtain a (possibly still not correct) result in much less time; for instance, this is commonly the case for similarity queries over multimedia data [25].
- For the same reason, the process of similarity search is typically iterative, because the user may be searching (using a feedback cycle [3]) for the "correct" query object or the "perfect" distance function for her current information needs. In early stages of this process, the user may just want to have a quick feel of what the data set contains.
- Finally, even when both the distance function and the query object are adequate, the user may still prefer to quickly obtain a (good enough) approximate result rather than to wait longer for the exact answer; for instance, if the user is driving her car and running out of fuel, getting as soon as possible information on the location of a close gas station could be preferred over waiting more time for the exact 1-NN.

The success of an approximate technique relies in solving the quality/time trade-off: Costs, measured as number of computed distance values and/or accessed disk pages for secondary memory structures, should be reduced as much as possible, while still keeping a high quality of the result. In the following, we denote the exact i-th NN of the query object q in X as $nn_X^i(q)$ and the i-th NN provided by an approximate algorithm as $\widetilde{nn}_X^i(q)$ (for simplicity, the algorithm is understood in the notation).

2. A Classification Schema

Based on the observation that exact similarity search is sometimes "difficult" (linear in the data set size), several approaches have been proposed to solve the approximate version of the problem. From an extensive analysis of the literature, we observed that virtually every approach formulates the approximate search problem in a new way, usually unrelated to prior techniques. With the aim to help comparing existing approaches, in this Section we introduce a schema able to classify them according to the following coordinates:

1. The type of space the approach applies to.
2. How approximation is obtained.
3. The guarantees on the result quality.
4. The degree of interaction with the user.

The above coordinates have been chosen in order to evaluate the field of applicability of existing techniques for approximate similarity search. In fact, it is understood that if a technique A is applicable only to a subset of the data to which another technique B is applicable, then A is less general than B. On the other hand, it could be the case that A is more efficient or leads to lower errors: We are not interested in overall efficiency or accuracy of existing techniques here, but only on how they are achieved and how they can be measured.

2.1. Data Type

The first dimension we propose classifies techniques based on the type of data they can be applied to. In this light, the following possibilities are considered, in increasing order of generality:

VS$_{L_p}$ (vector spaces, L_p distance) Techniques belonging to this class can only be applied when the considered objects are vectors in a D-dimensional space and the distance used to compare them is an L_p metric (thus no correlation between coordinates is allowed).[1] Specific classes can be obtained by instantiating p (e.g., the class **VS**$_{L_2}$ contains techniques that only apply to Euclidean spaces, i.e., when the distance used is the L_2 Euclidean metric). If p is not instantiated, then the technique is applicable to any vector space with an L_p metric, independently of the value of p.

VS (vector spaces) In this class fall all those techniques that explicitly use objects' coordinates (and are thus only applicable to vector spaces), but do not make any assumption on the distance used to compare vectors (thus arbitrary functions can be chosen, e.g., quadratic form functions where the distance between vectors is defined by way of a positive definite symmetric matrix [26]).

MS (metric spaces) Methods in this class are applicable to the more general case of objects drawn from an arbitrary metric space.

As examples of the above classification method, we now describe three approximations techniques, assigning each of them to the proper class.

Example 1. Locality-Sensitive Hashing (LSH) [17] transforms a D-dimensional vector p into a sequence of C bits

[1] We recall that the definition of the L_p distance between two points x and y in a D-dimensional space is as follows: $L_p(x,y) = \left(\sum_{i=1}^{D} |x[i]-y[i]|^p\right)^{1/p}$, $1 \le p < \infty$, $L_\infty(x,y) = \max_{i=1}^{D} |x[i]-y[i]|$.

(binary vector) $v(p)$. Since the L_1 distance between vectors can be approximated by the Hamming (edit) distance between the corresponding binary vectors, LSH uses an hashing technique to index only the binary vectors $v(p)$. Of course, both accuracy and efficiency of the technique highly depend on the number C of bits used for approximating vectors. Since approximation to the Hamming distance only yields for the L_1 metric, this technique is of class $\mathbf{VS}_{L_1}$.

Example 2. Approximate nearest neighbor search techniques based on the VA-file [31] are presented in [30]. The VA-file is a sequential structure containing approximations of vectors based on a fixed number b of bits. Exact k-NN search is performed by first executing a sequential scan of the structure using the distance on vectors approximations, which yields a number $M > k$ of *candidate bit vectors*, and then applying a refinement step, where the distance is evaluated on the corresponding real vectors and only the k closest ones are kept. The techniques in [30] either reduce the number of candidates by appropriately shrinking the query radius (VA-BND) or avoid the refinement phase at all, thus returning the closest k candidates (VA-LOW). Since no restriction is put on the distance to be used, both techniques fall in the $\mathbf{VS}$ class.

Example 3. The P-Sphere tree [18] is a 2-level index structure for approximate 1-NN search. In order to find the nearest neighbor of the query point, the leaf node closest to the query point is accessed. The query is solved through a simple linear scan of objects contained in such node. In this case, no assumption is made on the query distance to be used (which, however, should be the same used to build the tree) and no coordinates are used, thus this technique is classified as $\mathbf{MS}$.

2.2. Approximation Type

Our second classification dimension concerns how approximate techniques are able to reduce costs for similarity searches. The relevant cases to consider are:

$\mathbf{CS}$ (changing space) To this class belong approximate methods that first change the metric space, either by changing the distance used to compare objects or by modifying the object space, then solve the exact problem on the so-obtained approximate space, where the search is supposedly simpler. Examples of such techniques are those that approximate vectors using a fixed number of bits, or dimensionality reduction techniques.

$\mathbf{RC}$ (reducing comparisons) Techniques in this class use the exact distance to compare objects but reduce the number of objects to be compared against the query in order to obtain a speedup with respect to the exact search. This can be achieved by exploiting two different approaches (possibly both):

$\mathbf{RC}_{AP}$ (aggressive pruning) Regions of the metric spaces that are unlikely to contain results, but cannot be excluded from an exact search, are pruned by techniques in this class. Examples are those techniques that prune regions of the space using some probabilistic bounds.

$\mathbf{RC}_{ES}$ (early stopping) In this case, the search algorithm is terminated before correctness of the result can be proved. This is similar to an aggressive pruning of all the remaining objects/regions, but is usually performed without considering whether promising regions remain to be visited. Early stopping is commonly performed by expressing a maximum cost to be paid or an acceptable distance value to be reached.

Example 4. The VA-LOW technique discussed in Example 2 belongs to the $\mathbf{CS}$ class, since the approximate results are chosen by considering only their bit vector approximations.

Example 5. The BBD-tree [2] is a main memory index able to answer to approximate k-NN queries in a time that is poly-logarithmic in the number of objects in the data set.[2] To reduce the number of tree nodes accessed, during the search the query radius is reduced by a factor of ϵ with respect to the radius used for exact search. Therefore, this method can be classified as $\mathbf{RC}_{AP}$.

Example 6. Three different algorithms to solve approximate k-NN queries with M-tree [13] are presented in [32]. The first one reduces the current searching radius of the k-NN query by a factor of ϵ, thus it applies aggressive pruning ($\mathbf{RC}_{AP}$ class). Another technique employs the distance distribution to stop the search when the probability of finding a better result does not exceed a user-specified threshold, while the third technique simply interrupts the search when the improvement in the distance of the k-th NN falls below a threshold (the two latter approaches are in the $\mathbf{RC}_{ES}$ class).

Example 7. The technique proposed in [16] combines clustering and dimensionality reduction to approximate k-NN search. During the search, only the clusters which are closest to the query are considered and, for all the points in such clusters, only a fraction of dimensions is used to assess the distance to the query. To improve accuracy, the user can increase the number of visited clusters and/or the fraction of considered dimensions. This technique, therefore, combines characteristics of both classes $\mathbf{CS}$ (since only some

[2] However, the dependency on the space dimensionality D is exponential.

dimensions are used) and $\mathbf{RC}_{ES}$ (since only some clusters are explored).

2.3. Quality Guarantees

Having determined how approximate techniques are able to reduce costs, it is worth considering whether each method is able to bound from below the quality of its results. In other words, we are asking if an approximate technique can guarantee that its errors stay below a given value.[3] The classification we give is as follows:

NG (no guarantees) In this class fall all those methods that only use heuristic conditions to approximate the search; thus such methods are not able to give any formal bound on the error introduced by the approximation.

DG (deterministic guarantees) Techniques in this class are able to deterministically bound from above the error introduced by approximation.

PG (probabilistic guarantees) Approximate methods following this approach give probabilistic guarantees on the quality of query result. Usually this means that quality guarantees are met only for a given percentage ($< 100\%$) of the queries. To achieve this goal, information about distribution of data is needed. In this light, techniques belonging to this class can be further divided into two basic types according to how much it is known about objects' distribution [23].

- $\mathbf{PG}_{par}$ (parametric) Approaches in the parametric class assume that the data set follows a certain distribution; the only unknown information concerns a few parameters that need to be estimated (e.g., through sampling). Of course, when the considered objects do not follow the modeled distribution, quality guarantees cannot be met.

- $\mathbf{PG}_{npar}$ (non-parametric) In this case, little (or none at all) assumptions are made on the distribution of objects, so that such information has to be estimated and stored in a suitable way (e.g., using histograms).

Example 8. The third technique proposed in [32] (see Example 6), where the search is stopped when the distance improvement falls below an user-specified threshold, is in the $\mathbf{NG}$ class, because no guarantees can be given on the accuracy of the approximate result.

[3]In Section 5, we make more precise the issue of evaluating errors of approximate results.

Example 9. The algorithm for approximate search proposed for BBD-trees in [2] (see also Example 5), and the first technique proposed in [32] both use a value ϵ to reduce the query radius during the search. In both cases, it is guaranteed that the error, measured as $d\left(q, \widetilde{nn}_X^1(q)\right)/d\left(q, nn_X^1(q)\right) - 1$, cannot exceed ϵ, thus both techniques belong to class $\mathbf{DG}$.

Example 10. DBIN is a 2-levels index for solving the k-NN problem [4]. The method assumes that the data set is composed of K clusters, and that distribution of objects within each cluster can be modeled by way of a Gaussian distribution, parameterized by a mean vector and a covariance matrix. At query time, the cluster that best fits the query object is found, and the result is computed by considering objects in that cluster. Then, remaining clusters are accessed iff the probability that the k-NN have not been found yet is higher than a user-specified threshold. Such probability is computed by relying on the assumption of a Gaussian model, with parameters estimated at index construction time. Since the correct result is found only with high probability and a Gaussian distribution is assumed (where mean and covariance have to be estimated), DBIN is in the $\mathbf{PG}_{par}$ class.

Example 11. The PAC (Probably Approximately Correct) technique proposed in [12] is a paradigm for approximate 1-NN search with metric access methods, where the error (computed as in Example 9) is allowed to exceed the user-specified accuracy threshold ϵ with a probability limited by the user-specified confidence value δ. To guarantee this, the distance between the query objects and its 1-NN is estimated from the distance distribution [14] of indexed objects. Since this is not known at query time, it is estimated through sampling and stored in a histogram. By above considerations, this technique can be classified in the $\mathbf{PG}_{npar}$ class.

2.4. User Interaction

The last classification we propose relates to the possibility given to the user to specify, at query time, the parameters for the search (e.g., the maximum error allowed). Some techniques, in fact, are inherently static, in the sense that a structure is built by using a set of parameters to offer some guarantees: If the user wants to change, for example, the accuracy of the result, she has to modify the value of the parameters and to rebuild the structure from scratch. Other methods, on the other hand, exploit a single structure that is not bound to any parameter and can be used with different sets of parameters, according to current user's needs.

SA (static approach) When using a technique in this class, the user cannot freely choose the set of parameters for query approximation, but is bound to those specified when the (approximate) structure is built. Usually, to provide several quality of result profiles, different structures are built, using different sets of parameters, and the user is given the possibility to choose the structure that best fits her actual needs.

$\mathbf{IA}$ (interactive approach) Methods in this class are not bound to a specific set of parameters, but can be interactively used by varying such parameters at query time. Usually, interactive techniques are obtained as modifications of the exact similarity search method, which can be obtained by requesting a maximum error of 0%.

Example 12. In the P-Sphere technique presented in [18] (see Example 3), the size of leaf nodes, i.e., the number of objects in each data page, is estimated by taking into account a user-specified accuracy. Of course, if the accuracy parameter is changed, the P-Sphere tree has to be rebuilt from scratch. Therefore, this method is static and belongs to the $\mathbf{SA}$ class.

Example 13. The generalized NN search proposed in [19] is a new approach for high-dimensional NN search. The key idea here is to find (at query time) a suitable projection to reduce the space dimensionality; then, the NN search is performed on the reduced space using the original distance function and projected points. Of course, the higher the value of the dimensionality D' of the reduced space, the better accuracy is obtained by this technique. Since the user can specify, at query time, the value of D', this method can be classified as $\mathbf{IA}$.

3. Some Relevant Cases

In Table 1, the schema introduced in Section 2 is used to classify approaches for approximate similarity search presented in recent years. In order to appreciate how the schema can contribute to synthetically characterize existing approaches, in the following we discuss a few of them in more detail.

FastMap [15]: $(\mathbf{MS}, \mathbf{CS}, \mathbf{NG}, \mathbf{SA})$ The FastMap technique [15] has been proposed as a tool for mining and visualizing metric data sets. In its essence, the FastMap algorithm is able to map a set of objects drawn from a generic metric space to a D'-dimensional Euclidean space, where D' is a user-specified value, such that distances between objects are preserved as much as possible. Of course, this approach can also be used for approximate searching, since performing a similarity search in the target D'-dimensional space can be viewed as an approximate search in the original metric space. Since the method applies to general metric spaces, it belongs to the $\mathbf{MS}$ class; the transformation of the space leads to a transformation of the distance used to compare objects, thus this technique is in the $\mathbf{CS}$ class; in the paper, the authors give no guarantee on the error introduced for distance in the target space,[4] hence the quality guarantee class is $\mathbf{NG}$; finally, as for user interaction, the mapping in the D'-dimensional space has to be performed before any index structure is built on the transformed objects, thus FastMap falls in the $\mathbf{SA}$ class.

DBIN [4]: $(\mathbf{VS}, \mathbf{RC}_{ES}, \mathbf{PG}_{par}, \mathbf{IA})$ The DBIN (density based indexing) method was presented in [4] as an approach to solve approximate similarity queries in high-dimensional spaces. The basic assumption is that the distribution of objects in the space can be modeled as a mixture of Gaussian distributions. Each point, therefore, can be associated to a cluster, parameterized with a mean vector and a covariance matrix, by using an expectation-maximization algorithm. When searching for the NN of a query point, the clusters obtained in the building phase are ranked according to the probability that the query point belongs to them; then, each cluster is accessed (and points in that cluster compared to the query) until the probability that the NN has not been found falls below an user-specified tolerance. Since no assumption is made on the distance used to compare vectors (even if analytical results are given only in the case of quadratic form distance functions), this method falls in class $\mathbf{VS}$; the search is early terminated by using data distribution, thus the class of this technique is $\mathbf{RC}_{ES}$; as for quality guarantees, this technique assumes that indexed objects are distributed in clusters according to a Gaussian distribution, for which the mean and the covariance are estimated in the building phase, hence this method belongs to class $\mathbf{PG}_{par}$; since the user can specify the tolerance parameter, used when stopping the search, this method is in class $\mathbf{IA}$.

PAC [12]: $(\mathbf{MS}, \mathbf{RC}, \mathbf{PG}_{npar}, \mathbf{IA})$ PAC (probably approximately correct) nearest neighbor queries, introduced in [12], represent a probabilistic approach to approximate 1-NN search in metric spaces, where the error in the result can exceed a specified accuracy threshold ϵ with a probability that is limited by a confidence parameter δ. The PAC paradigm can be applied to *any* distance-based (either multi-dimensional or metric) index tree that is based on a recursive and conservative decomposition of the space (thus, it is in $\mathbf{MS}$ class). The only information that is needed by the algorithm to prune index nodes from the search is the value of $r_\delta(q)$, the maximum value of distance from the query object q for which the probability that the exact NN

[4]The error between exact distances and distances between transformed objects can be limited, for the relevant case of vector spaces, by exploiting the Johnson-Lindenstrauss lemma. However, for the general case of metric spaces, no general rule has been proposed so far.

name	data type	approximation type	quality guarantees	user interaction
LSH [17]	$\mathbf{VS}_{L_1}$	$\mathbf{CS}$	$\mathbf{PG}_{npar}$	$\mathbf{SA}$
Clustering+Precision [6]	$\mathbf{VS}_{L_2}$	$\mathbf{RC}_{AP}$	$\mathbf{PG}_{npar}$	$\mathbf{SA}$
AB-tree [24]	$\mathbf{VS}_{L_2}$	$\mathbf{RC}_{AP}$	$\mathbf{PG}_{par}$	$\mathbf{IA}$
BBD-tree [2]	$\mathbf{VS}_{L_p}$	$\mathbf{RC}_{AP}$	$\mathbf{DG}$	$\mathbf{IA}$
VA-LOW [30]	$\mathbf{VS}$	$\mathbf{CS}$	$\mathbf{DG}$	$\mathbf{SA}$
DBIN [4]	$\mathbf{VS}$	$\mathbf{RC}_{ES}$	$\mathbf{PG}_{par}$	$\mathbf{IA}$
Generalized Search [19]	$\mathbf{VS}$	$\mathbf{CS}$	$\mathbf{DG}$	$\mathbf{IA}$
VA-BND [30]	$\mathbf{VS}$	$\mathbf{RC}_{AP}$	$\mathbf{PG}_{npar}$	$\mathbf{IA}$
Integrated Progressive Search [16]	$\mathbf{VS}$	$\mathbf{CS}$, $\mathbf{RC}_{ES}$	$\mathbf{NG}$	$\mathbf{IA}$
Clindex [22]	$\mathbf{VS}$	$\mathbf{RC}_{ES}$	$\mathbf{NG}$	$\mathbf{IA}$
VQ-index [27]	$\mathbf{VS}$	$\mathbf{CS}$, $\mathbf{RC}_{ES}$	$\mathbf{NG}$	$\mathbf{IA}$
Buoy indexing [28]	$\mathbf{VS}$	$\mathbf{RC}_{ES}$	$\mathbf{NG}$	$\mathbf{IA}$
CSVD [9]	$\mathbf{VS}$	$\mathbf{CS}$	$\mathbf{NG}$	$\mathbf{IA}$
FastMap [15]	$\mathbf{MS}$	$\mathbf{CS}$	$\mathbf{NG}$	$\mathbf{SA}$
MetricMap [29]	$\mathbf{MS}$	$\mathbf{CS}$	$\mathbf{NG}$	$\mathbf{SA}$
P-Sphere tree [18]	$\mathbf{MS}$	$\mathbf{RC}_{AP}$	$\mathbf{PG}_{npar}$	$\mathbf{SA}$
M-tree: Relative Error [32]	$\mathbf{MS}$	$\mathbf{RC}_{AP}$	$\mathbf{DG}$	$\mathbf{IA}$
M-tree: Good Fraction [32]	$\mathbf{MS}$	$\mathbf{RC}_{ES}$	$\mathbf{NG}$	$\mathbf{IA}$
M-tree: Improvement Slowdown [32]	$\mathbf{MS}$	$\mathbf{RC}_{ES}$	$\mathbf{NG}$	$\mathbf{IA}$
PAC [12]	$\mathbf{MS}$	$\mathbf{RC}$	$\mathbf{PG}_{npar}$	$\mathbf{IA}$
Distinctive NN [21]	$\mathbf{MS}$	$\mathbf{RC}_{ES}$	$\mathbf{PG}_{npar}$	$\mathbf{IA}$
Probabilistic Proximity Search [10, 11]	$\mathbf{MS}$	$\mathbf{RC}_{AP}$	$\mathbf{PG}_{npar}$	$\mathbf{IA}$
Proximity-based [1]	$\mathbf{MS}$	$\mathbf{RC}_{AP}$	$\mathbf{DG}$	$\mathbf{IA}$
Probabilistic Incremental Search [8]	$\mathbf{MS}$	$\mathbf{RC}_{ES}$	$\mathbf{NG}$	$\mathbf{IA}$
Genetic Search [7]	$\mathbf{MS}$	$\mathbf{RC}$	$\mathbf{NG}$	$\mathbf{IA}$

Table 1. Classification of approaches for approximate similarity search.

of q has a distance lower than r is not greater than δ:

$$r_\delta(q) = \sup\{r | \Pr\{d\left(q, nn_X^1(q)\right) \le r\} \le \delta\} \qquad (1)$$

In the paper, this value is estimated by using the distance distribution of indexed objects with respect to the query object (obtained through sampling and stored as an histogram); it is therefore clear that this approach is probabilistic and non-parametric ($\mathbf{PG}_{npar}$ class). The distance used to query the distance-based index structure is the exact one, the approximation is introduced by reducing the number of object to be compared against the query object q by means of $r_\delta(q)$ and of the ϵ parameter, thus the class of this approach is $\mathbf{RC}$ (actually, both $\mathbf{RC}_{AP}$ and $\mathbf{RC}_{ES}$); finally, since the accuracy and the confidence parameters (ϵ and δ, respectively) can be specified at query time, this technique belongs to the $\mathbf{IA}$ class.

VA-BND [30]: $(\mathbf{VS}, \mathbf{RC}_{AP}, \mathbf{PG}_{npar}, \mathbf{IA})$ Two approximate query evaluation techniques are presented in [30] for the VA-File. The VA-File structure [31] approximates vectors using a fixed number of bits, and stores such approximations in a file. For exact k-NN search, the approximation file is sequentially scanned to exclude vectors that cannot be in the result set through the computation of bounds on exact distances (such scan is very fast since the computation of bounds between approximations has to consider only a few bits); finally, exact vectors corresponding to approximations included in the result of the previous scan (the "candidate bit vectors") are compared against the query point to compute the final result. Since the approximations of the VA-File are only applicable to vector spaces and any distance can be used to compare vectors (even if computation of bounds can be a difficult task if complex metrics are used), all approximate techniques developed for this structure fall in the $\mathbf{VS}$ class.

The first approach to reduce the complexity of similarity searching in the VA-File through approximation proposes to adapt the computation of distance between approximate vectors. The user is given the possibility to specify a value α to adapt computed bounds: Higher values of α correspond to higher errors in the result, but the candidate set will consist in a lower number of vectors. Since the approximation is introduced in the computation of bounds and not on the exact distance, this technique can be classified as $\mathbf{RC}_{AP}$. The number of vectors missed can be computed as a function of the distance distribution between objects, thus this technique can give probabilistic guarantees as a function of the parameter α; therefore, the class for this method is $\mathbf{PG}_{npar}$. Finally, since the parameter α can be specified at query time, the VA-BND technique is in class $\mathbf{IA}$.

VA-LOW [30]: $(\mathbf{VS}, \mathbf{CS}, \mathbf{DG}, \mathbf{SA})$ The second approximate technique for the VA-file (also presented in [30]) completely omits the refinement phase and returns, as the approximate result, the k vectors corresponding to the best candidate bit vectors. Since in this case errors in the result arise from using the approximate vectors instead of the exact ones, this method can be classified as $\mathbf{CS}$. The error can be controlled by means of the quantity of bits used for the approximations: The more bits are used, the better the approximation but the slower the sequential scan. Since a bound on the error between the distance on approximate vectors and the exact distance can be easily computed, this technique falls in the $\mathbf{DG}$ class. As for the interaction with the user, it is clear that the only parameter used, i.e., the number of bits used for vectors' approximation, has to be specified before the actual VA-File is built, so that the class for this technique is $\mathbf{SA}$.

Probabilistic Proximity Search [11]: $(\mathbf{MS}, \mathbf{RC}_{AP}, \mathbf{PG}_{npar}, \mathbf{IA})$ The technique described in [10, 11] is basically an adaptation of search radius reduction to pivot-based searching algorithms.[5] The novelty here is that the reduction of the search radius is not specified by the user, but calculated by using the (inverse of the) distance distribution so as to provide a probabilistic guarantee on the approximate result. This can be classified as follows: $\mathbf{MS}$ (because pivot-based algorithms are applicable to generic metric spaces), $\mathbf{RC}_{AP}$ (aggressive pruning is used by reducing the search radius), $\mathbf{PG}_{npar}$ (guarantees on the result quality are probabilistic and non parametric), and $\mathbf{IA}$ (a regular pivot-based index is used and the confidence level can be expressed by the user at query time).

3.1. Comments and Extensions

We believe that the proposed classification schema can be very fruitful for the analysis of approximate techniques for similarity search. By using such schema interesting relations and similarities between techniques can be found that may not be evident at a first sight. As an example, consider the PAC and the VA-BND techniques: Both are classified as belonging to the $\mathbf{RC}$, $\mathbf{PG}_{npar}$, and $\mathbf{IA}$ classes, the only difference being in the fact that VA-BND only applies to vector spaces. Indeed, at a closer look, these two method share several analogies:

- In both cases the approach requests for an additional parameter (ϵ and α, respectively) representing the quality of the result the user is willing to obtain. The lower the value of the parameter, the lower the error and the higher the search costs.
- Both methods use information about the distance distribution in order to estimate the distance between the query object and its nearest neighbor.
- In both cases the distance distribution and the parameter are jointly used to derive bounds to stop the search.

In the same way, one could discover that similarities exist between these two approaches and Probabilistic Proximity Search.

It is clear that, by using the proposed classification schema, we are able to immediately understand the field of applicability of a particular approximate technique. In this way, we can conceive whether, for example, a method is more general, i.e., it applies to a superset of scenarios with respect to another, or how its quality measures relate to those proposed for other techniques. In search for the "best" approximate technique for a specific scenario at hand, in fact, different aspects are to be considered, in particular the generality/efficiency trade-off: A more general method is expected to have a lower efficiency (i.e., to lead to higher search costs or to worse quality) with respect to a method that applies to a lower number of cases; for example, this could apply when considering methods for metric spaces or just for vector spaces. The same considerations can be made when dealing with quality guarantees: Parametric approaches usually attain better performance with respect to non-parametric ones, yet they are only applicable to particular distributions of objects. On the other hand, deterministic techniques provide stronger quality guarantees than probabilistic ones, yet they usually incur higher costs [12]. Finally, it is clear that interactive approaches are more general than static ones, since the user is given the possibility to choose at query time the desired quality of the result, which is inversely related to search costs needed to obtain the approximate result.

4. Optimal Approximate Similarity Search

Having discussed how several coordinates can help in better understanding the scope of an approximate similarity technique, now we turn back to the basic problem that

[5] Although only the range search algorithm is given by the authors, the technique can be easily extended to k-NN queries.

any technique has to face, i.e., optimizing the quality/time trade-off. As seen, this ultimate goal can be approached in several ways. To start simple, here we concentrate on $\mathbf{RC}_{ES}$ techniques, thus assuming that cost reduction is obtained by stopping earlier with respect to an exact search. Further, we assume that early stopping is the *only* difference with respect to exact search (i.e., the original object space is considered and no aggressive pruning technique is applied). As a consequence, we view the problem as an *on-line* process, in which the exact result is eventually reachable if enough time is allocated. Besides the intrinsic attractiveness of an on-line scenario, in which the quality of results can be improved over time, this allows approximate search to be viewed as a *generalization* of exact one. Indeed, the user has the possibility to suspend the search process by means of a specific stopping condition (e.g., minimum distance or maximum cost), and to resume it (without starting from scratch) if she is unsatisfied with the current result.

Given the above, it is therefore natural to ask whether, given a stopping condition, an approach is able to provide the best possible result, i.e., the least cost for reaching a certain distance threshold θ or the minimum distance after a given cost c has been paid. Surprisingly enough, very few approaches consider this important problem. In the following, we provide optimality results for on-line approximate queries, i.e., we show how it is possible to obtain the best results as early as possible.

We consider an 1-NN search over a 2-levels index that organizes objects in X as a collection of n nodes (nodes can contain just 1 object, so flat structures are included in this description), since this is the scenario more amenable to be formally characterized. For each node N_i, a compact representation of N_i is provided by the tree (for example, the center x_i and the radius r_i for a ball-partitioning index, like M-tree [13]). The search algorithm accesses the nodes according to a *scheduling policy* Π until the stopping condition is met or all nodes have been accessed (or pruned). The schedule Π can thus be viewed as a permutation of the set $\{1, \ldots, n\}$:

$$\Pi = (\Pi_1, \Pi_2, \ldots, \Pi_i, \ldots, \Pi_n)$$

where N_{Π_i} is the leaf that schedule Π will fetch at step i. In order to build Π, information about the query q and the compact representation of each node N_i provided by the tree are used: For example, the MinDist policy [5, 20] orders nodes for increasing values of the lower bound of the distance between q and any object in N_i (this equals $\max\{d(q, x_i) - r_i, 0\}$ for a ball-partitioning index). We suppose, as is commonly the case, that the position of each node N_i in the schedule Π depends on the query and on the statistics of N_i provided by the tree, thus Π does not change during the search. In the case scheduling of nodes also depends on the current approximate result, the order of nodes should be dynamically adjusted each time a better approximate result is found.[6]

To start with, we precisely define what an optimal schedule for approximate search is. In order to compare two schedules one can either fix the maximum distance, θ, and look at the costs paid to get such quality, or, going the other way, compare the results obtained when a certain cost c is paid.

Definition 1 (Optimal Schedule for Query q)**.** *Let $\mathcal{T}$ be an index tree over a data set $X \subset \Omega$, and $q \in \Omega$ a query point.*

We say that schedule Π cost-dominates at distance level θ *schedule* Π' *iff $Cost(q; \Pi, \theta) < Cost(q; \Pi', \theta)$, where $Cost(q; \Pi, \theta)$ measures the (minimum) cost paid for query q to return a result with distance $\leq \theta$ adopting* Π. *Schedule* Π *is* cost-optimal for q at distance level θ *iff there exists no schedule Π' that cost-dominates Π for that query at level θ and is* cost-optimal for q *iff it is cost-optimal for q at all distance levels.*

Similarly, schedule Π distance-dominates at cost level c *schedule Π' iff $d(q; \Pi, c) < d(q; \Pi', c)$, where $d(q; \Pi, c)$ is the distance of the 1-NN for query q when the search pays a cost $\leq c$ adopting Π. Schedule Π is* distance-optimal for q at cost level c *iff there exists no schedule Π' that distance-dominates Π for that query at level c and is* distance-optimal for q *iff it is distance-optimal for q at all cost levels.*

Example 14. Consider the two schedules in Figure 1: Schedule Π_1 cost-dominates Π_2 at distance level θ_1, whereas the opposite is true at distance level θ_2; also, Π_1 distance-dominates Π_2 at cost level c_1 and is dominated by Π_2 at cost level c_2.

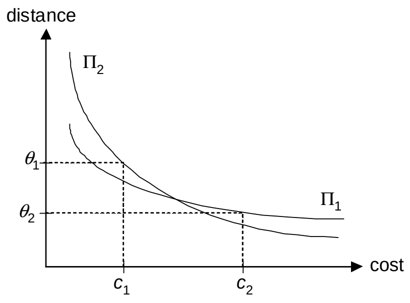

Figure 1. Comparing two schedules.

Our first result states that the two notions of optimality in Definition 1 coincide.

Lemma 1. *A schedule Π is cost-optimal for q iff it is distance-optimal for q.*

[6]This happens with the scheduling proposed in [1], which however incurs high costs for managing the queue of nodes and yields almost the same results as a fixed schedule (G. Amato: personal communication).

Proof. Let Π be distance-optimal for q. Assume by contradiction that Π is not cost-optimal, i.e., there exists a distance level θ' and a schedule Π' such that $c' = Cost(q; \Pi', \theta') < c = Cost(q; \Pi, \theta')$. By definition of cost, for any value less than c the distance obtainable from Π is higher than θ'. It follows that the distance of Π at cost level $c' < c$ is larger than θ'. This contradicts the hypothesis that Π is distance-optimal. Proving that cost-optimality implies distance-optimality follows similar arguments. □

Because of the above lemma, it is possible to just say that a schedule is optimal for q. Unfortunately, we have the following negative result.

Lemma 2. *For any query q there exists no optimal schedule for q.*

Proof. First observe that in no case we can completely determine the content of a node N_i (and, in particular, the distance value of the closest point to q in N_i) before accessing N_i. Consider now two distinct data sets, X_A and X_B, leading respectively to index trees $\mathcal{T}_A$ and $\mathcal{T}_B$, each having just two leaf nodes (N_{A1} and N_{A2}, N_{B1} and N_{B2}, respectively). The NN of q is in N_{A1} for X_A and in N_{B2} for X_B. Assume also that, from the query q point of view, the statistics of N_{A1} equal those of N_{B1} and the statistics of N_{A2} equal those of N_{B2}, i.e., q in both cases would "see" the same tree. Then, any scheduling policy will act the same on both trees. Assume Π is an optimal schedule for q and, without loss of generality, that Π first fetches node N_{A1} in $\mathcal{T}_A$ and N_{B1} in $\mathcal{T}_B$. Consider now Π' that orders nodes in the opposite way. Since Π' performs better than Π on $\mathcal{T}_B$, this contradicts the hypothesis that Π is optimal. □

Arguments used in the proof of above lemma can also be applied to show that neither cost-optimality nor distance-optimality can be obtained for any distance/cost level. It is also interesting to observe that this holds even for the 0 error case, in spite of the I/O-optimality (proved in [5, 20]) of the MINDIST policy that orders nodes by increasing values of the lower bound on distances between the query q and objects in each node N_i. Indeed, although MINDIST minimizes the number of I/Os of any *provable* exact search, this does not imply that MINDIST is the schedule that can return earlier the exact result in an online search. As also observed in [12], the hard problem in exact k-NN search is not locating the nearest neighbors, but reaching the stop condition that guarantees correctness of the result.

4.1. Optimal-on-the-Average Schedules

Because of above results, one needs to somewhat relax the optimality requirements. Therefore, here we consider schedules that, although not necessarily optimal for a query, are *optimal-on-the-average*. A schedule Π that is either cost- or distance-optimal-on-the-average has the property that no other schedule Π' performs better than it when a random query is considered.

Theorem 1. *The cost- and distance-optimal-on-the-average schedules are incrementally obtained by choosing at each step j the node that, among the unread nodes N_i, maximizes the quantities listed in Table 2 (where $G_i(r)$ is the distance distribution of the 1-NN of q in node N_i, $G_i(r) = \Pr\{d(q, nn^1_{N_i}(q)) \le r\}$, and d^+ denotes the maximum distance value for d).*

Type of optimality	cost-optimal	distance-optimal
Scenario	given θ, minimize avg. cost	given c, minimize avg. distance
Quantity to maximize at each step	$G_i(\theta)$	$\int_0^{d^+} G_i(r)\ dr$

Table 2. Deriving optimal-on-the-average-schedules.

Proof. The formal proof requires the precise definition of a cost model for on-line approximate search. Due to space limitation, we only offer an intuitive sketch: A cost-optimal schedule should choose, at each step, the node maximizing the probability of finding a point whose distance to q is not higher than θ, i.e., the one maximizing $G_i(\theta)$. On the other hand, a distance-optimal schedule has to minimize the distance to the 1-NN at each step, thus one should choose the node that minimizes the expected 1-NN distance to the query, which can be expressed as [14]:

$$E\left[d\left(q, nn^1_{N_i}(q)\right)\right] = d^+ - \int_0^{d^+} G_i(r)\ dr$$

□

To conclude, we discuss approaches that take into account the node scheduling, showing how they can be viewed as a special case of our results. We first consider the MINDIST schedule which, as we saw, is optimal for exact k-NN search [5, 20]: If we take the optimistic case, where all the mass probability $G_i()$ of node N_i is concentrated in the MINDIST value between N_i and q, then ordering on MINDIST values indeed coincides with minimizing the expected 1-NN distance. In the DBIN approach [4], nodes are ordered for decreasing values of the probability to find a better result: This is the same as maximizing the value of $G_i(r)$, where r is the distance between q and its current 1-NN (note that DBIN is *not* an on-line algorithm). Finally, we note that the importance of node scheduling has been first addressed in [8], where an on-line algorithm is proposed. However, no formal result is given and used strategies are only based on heuristic considerations.

5. Evaluating the Quality of Results

A fundamental problem regarding the evaluation of approximate searching algorithms is how the quality of attained results is assessed. This is commonly obtained by comparing the results of approximate and exact algorithms. In this particular aspect, the lack of a common framework for the definition of approximate search techniques is plainly manifest, especially for the case of k-NN queries with $k > 1$. In this case, in fact, virtually every technique published in literature proposes its own definition of result quality. All such measures, however, conform to one of the following general families:

***Ranking*-based** This class contains those measures that use comparisons in ranking of objects between approximate and exact results. To this end, the function $rank(p)$ is used to return the rank of object p in the exact result, i.e., if p is the i-th NN of q in X, it is $rank(p) = i$.

***Distance*-based** In this case, the quality of result is defined by comparing the distance to the query of exact and approximate results.

Clearly, in order to compute the rank of a point the correct ranked list of all the objects of X has to be provided, thus ranking-based measures are usually very expensive to compute. Examples of performance measures are included in Table 3.

Limits of above-described quality measures are highlighted by the following example.

Example 15. In Table 4 we consider the result provided by an approximate algorithm for four different queries over a data set X of cardinality N, where the distance between the queries and its 1-NN is constant, $d\left(q_i, nn_X^1\left(q_i\right)\right) = 1, \forall i$.

	q_1	q_2	q_3	q_4
$d\left(q_i, \widetilde{nn}_X^1\left(q_i\right)\right)$	2	2	100	100
$rank(\widetilde{nn}_X^1\left(q_i\right))$	2	100	2	100

Table 4. Results provided by an approximate algorithm for four different queries.

By comparing results over the different queries, it is clear that the highest quality is obtained for q_1, since the error on distance is low, as is the difference in ranking. The worst result is obtained for q_4, having highest errors in both distance and ranking, whereas the quality for q_2 and q_3 is similar and lies in between such extreme values. However, if we only consider a ranking-based measure, q_3 has the same quality (the best) as q_1, whereas, with a distance-based measure, q_3 shares the worst quality value with q_4. In a similar way, with ranking-based (respectively, distance-based) measures, q_2 has the worst (resp. best) result.

Considerations included in Example 15 show that a "good" definition for the quality of an approximate result should take into account *both* the ranking *and* the distance criteria. A natural choice is to combine the two measures in order to obtain a value in the interval $[0, 1]$, with values close to 1 indicating a result close to the exact one. A possible definition of quality for a 1-NN approximate query is therefore the following:

$$\mathcal{Q} = \left(1 - \frac{rank(\widetilde{nn}_X^1\left(q\right)) - 1}{|X|}\right) \cdot \frac{d\left(q, nn_X^1\left(q\right)\right)}{d\left(q, \widetilde{nn}_X^1\left(q\right)\right)} \quad (2)$$

Using such definition it is immediate to compute the quality for the four data sets considered in Example 15. By referring to values included in Table 4, it is:

	q_1	q_2	q_3	q_4
$\mathcal{Q}$	$\frac{N-1}{N} \cdot \frac{1}{2}$	$\frac{N-99}{N} \cdot \frac{1}{2}$	$\frac{N-1}{N} \cdot \frac{1}{100}$	$\frac{N-99}{N} \cdot \frac{1}{100}$

Equation 2 can be generalized to $\mathcal{Q}_i = \left(1 - \frac{rank(\widetilde{nn}_X^i(q))-i}{|X|}\right) \cdot \frac{d\left(q, nn_X^i(q)\right)}{d\left(q, \widetilde{nn}_X^i(q)\right)}$ for the i-th NN. The overall quality for an approximate algorithm over a k-NN query can then be assessed by combining the $\mathcal{Q}_i$ values, e.g., by averaging them: $\mathcal{Q} = \frac{1}{k}\sum_{i=1}^{k} \mathcal{Q}_i$.

As a final observation, it is important to bear in mind that specific application requirements might favor a quality measure over other ones. For instance, in the simple example in Section 1.1 in which the problem is to obtain as soon as possible information on a close enough gas station, it is evident that ranking-based measures are inappropriate, since the user is only concerned with the distance to be traveled.

6. Conclusions

In this paper we have introduced a schema for classifying approaches to the approximate similarity search problem and have shown how many techniques that appeared in the literature can be consequently organized. In perspective, we believe that such schema can be profitably used by other researchers. We have also discussed the relevance of two, often underestimated, facets of the approximate search problem, namely scheduling of index nodes and measures

measure	definition	type	range (worst,best)	used in	notes
Precision/recall	$p = \frac{1}{k} \cdot \sum_{i=1}^{k} \begin{cases} 0 & \text{if } rank(\widetilde{nn}_X^i(q)) > k \\ 1 & \text{otherwise} \end{cases}$	ranking	(0,1)	[2, 28, 9, 1, 8, 29]	
Normalized rank sum	$nrs = \frac{k(k+1)}{2 \cdot \sum_{i=1}^{k} rank(\widetilde{nn}_X^i(q))}$	ranking	(0,1)	[30]	
Ratio of false dismissals	$rfd = \frac{1}{k} \cdot \sum_{i=1}^{k} \begin{cases} 1 & \text{if } rank(\widetilde{nn}_X^i(q)) > k \\ 0 & \text{otherwise} \end{cases}$	ranking	(1,0)	[4, 30, 6]	$rfd = 1 - p$
Error on position	$EP = \frac{\sum_{i=1}^{k} rank(\widetilde{nn}_X^i(q)) - i}{k \cdot \lvert X \rvert}$	ranking	(1,0)	[1]	
Effective error	$\epsilon_{\mathit{eff}} = \frac{d(q, \widetilde{nn}_X^1(q))}{d(q, nn_X^1(q))} - 1$	distance	(∞,0)	[2, 17, 12, 24, 11]	
Relative distance error	$\overline{\epsilon} = \frac{1}{k} \cdot \sum_{i=1}^{k} \left(\frac{d(q, \widetilde{nn}_X^i(q))}{d(q, nn_X^i(q))} - 1 \right)$	distance	(∞,0)	[32]	avg. of ϵ_{eff}
Total distance ratio	$DR = \frac{\sum_{i=1}^{k} d(q, nn_X^i(q))}{\sum_{i=1}^{k} d(q, \widetilde{nn}_X^i(q))}$	distance	(0,1)	[16, 27]	

Table 3. Performance measures for approximate similarity search.

for assessing the quality of the results. Even if limited to a specific class of approximation techniques, our formal result on optimality of schedules has to be viewed as starting point upon which more general theories could be built in the future. Finally, we would like to stress the importance of using significant quality measures when comparing different approximate techniques; in particular, specific needs of the domain at hand should be the guide in the quest towards the best quality measure.

References

[1] G. Amato, F. Rabitti, P. Savino, and P. Zezula. Region proximity in metric spaces and its use for approximate similarity search. *ACM Transactions on Information Systems*, 21(2):192–227, 2003.

[2] S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, and A. Y. Wu. An optimal algorithm for approximate nearest neighbor searching. *Journal of the ACM*, 45(6):891–923, Nov. 1998.

[3] I. Bartolini, P. Ciaccia, and F. Waas. FeedbackBypass: A new approach to interactive similarity query processing. In *Proceedings of the 27th International Conference on Very Large Data Bases (VLDB 2001)*, pages 201–210, Rome, Italy, Sept. 2001. Morgan Kaufmann.

[4] K. P. Bennett, U. M. Fayyad, and D. Geiger. Density-based indexing for approximate nearest-neighbor queries. In *Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 1999)*, pages 233–243, San Diego, CA, Aug. 1999. ACM Press.

[5] S. Berchtold, C. Böhm, D. A. Keim, and H.-P. Kriegel. A cost model for nearest neighbor search in high-dimensional data space. In *Proceedings of the 16th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS'97)*, pages 78–86, Tucson, AZ, May 1997. ACM Press.

[6] S.-A. Berrani, L. Amsaleg, and P. Gros. Approximate searches: k-neighbors + precision. In *Proceedings of the 2003 ACM CIKM International Conference on Information and Knowledge Management*, pages 24–31, New Orleans, LA, Nov. 2003. ACM Press.

[7] R. Bueno, A. J. M. Traina, and C. Traina, Jr. Genetic algorithms for approximate similarity queries. *Data and Knowledge Engineering*, 62(3):459–482, Sept. 2007. MS, RCEP, NG, IA.

[8] B. Bustos and G. Navarro. Probabilistic proximity searching algorithms based on compact partitions. *Journal of Discrete Algorithms*, 2(1):115–134, 2004.

[9] V. Castelli, A. Thomasian, and C.-S. Li. CSVD: Clustering and singular value decomposition for approximate similarity search in high-dimensional spaces. *IEEE Transactions on Knowledge and Data Engineering*, 15(3):671–685, 2003.

[10] E. Chávez and G. Navarro. A probabilistic spell for the curse of dimensionality. In *Proceedings of the 3rd International Workshop on Algorithm Engineering and Experimentation (ALENEX 2001)*, volume 2153 of *Lecture Notes in Computer Science*, pages 147–160, Washington, DC, Jan. 2001. Springer.

[11] E. Chávez and G. Navarro. Probabilistic proximity search: Fighting the curse of dimensionality in metric spaces. *Information Processing Letters*, 85(1):39–46, Jan. 2003.

[12] P. Ciaccia and M. Patella. PAC nearest neighbor queries: Approximate and controlled search in high-dimensional and metric spaces. In *Proceedings of the 16th International Conference on Data Engineering (ICDE 2000)*, pages 244–255, San Diego, CA, Mar. 2000. IEEE Computer Society.

[13] P. Ciaccia, M. Patella, and P. Zezula. M-tree: An efficient access method for similarity search in metric spaces. In *Proceedings of the 23rd International Conference on Very Large Data Bases (VLDB'97)*, pages 426–435, Athens, Greece, Aug. 1997. Morgan Kaufmann.

[14] P. Ciaccia, M. Patella, and P. Zezula. A cost model for similarity queries in metric spaces. In *Proceedings of the 16th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS'98)*, pages 59–68, Seattle, WA, June 1998. ACM Press.

[15] C. Faloutsos and K.-I. Lin. FastMap: A fast algorithm for indexing, data-mining and visualization of traditional and

multimedia datasets. In *Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data*, pages 163–174, San Jose, CA, June 1995. ACM Press.

[16] H. Ferhatosmanoglu, E. Tuncel, D. Agrawal, and A. El Abbadi. Approximate nearest neighbor searching in multimedia databases. In *Proceedings of the 17th International Conference on Data Engineering (ICDE 2001)*, pages 503–511, Heidelberg, Germany, Apr. 2001. IEEE Computer Society.

[17] A. Gionis, P. Indyk, and R. Motwani. Similarity search in high dimensions via hashing. In *Proceedings of 25th International Conference on Very Large Data Bases (VLDB'99)*, pages 518–529, Edinburgh, Scotland, UK, Sept. 1999. Morgan Kaufmann.

[18] J. Goldstein and R. Ramakrishnan. Contrast plots and P-Sphere trees: Space vs. time in nearest neighbor searches. In *Proceedings of 26th International Conference on Very Large Data Bases (VLDB 2000)*, pages 429–440, Cairo, Egypt, Sept. 2000. Morgan Kaufmann.

[19] A. Hinneburg, C. C. Aggarwal, and D. A. Keim. What is the nearest neighbor in high dimensional spaces? In *Proceedings of 26th International Conference on Very Large Data Bases (VLDB 2000)*, pages 506–515, Cairo, Egypt, Sept. 2000. Morgan Kaufmann.

[20] G. R. Hjaltason and H. Samet. Index-driven similarity search in metric spaces. *ACM Transactions on Database Systems*, 28(4):517–580, Dec. 2003.

[21] N. Katayama and S. Satoh. Distinctiveness-sensitive nearest neighbor search for efficient similarity retrieval of multimedia information. In *Proceedings of the 17th International Conference on Data Engineering (ICDE 2001)*, pages 493–502, Heidelberg, Germany, Apr. 2001. IEEE Computer Society.

[22] C. Li, E. Y. Chang, H. Garcia-Molina, and G. Wiederhold. Clustering for approximate similarity search in high-dimensional spaces. *IEEE Transactions on Knowledge and Data Engineering*, 14(4):792–808, 2002.

[23] M. V. Mannino, P. Chu, and T. Sager. Statistical profile estimation in database systems. *ACM Computing Surveys*, 20(3):191–221, Sept. 1988.

[24] S. Pramanik, J. Li, and J. Ruan. Performance analysis of AB-tree. In *Proceedings of the 2000 IEEE International Conference on Multimedia and Expo (ICME)*, volume III, pages 1701–1704, New York, NY, Aug. 2000. IEEE Computer Society.

[25] S. Santini and R. Jain. Beyond query by example. In *Proceedings of the 6th ACM International Conference on Multimedia '98*, pages 345–350, Bristol, UK, Sept. 1998. ACM Press.

[26] T. Seidl and H.-P. Kriegel. Efficient user-adaptable similarity search in large multimedia databases. In *Proceedings of the 23rd International Conference on Very Large Data Bases (VLDB'97)*, pages 506–515, Athens, Greece, Aug. 1997. Morgan Kaufmann.

[27] E. Tuncel, H. Ferhatosmanoglu, and K. Rose. VQ-index: an index structure for similarity searching in multimedia databases. In *Proceedings of the 10th ACM International Conference on Multimedia 2002*, pages 543–552, Juan les Pins, France, Dec. 2002. ACM Press.

[28] S. Volmer. Fast approximate nearest-neighbor queries in metric feature spaces by buoy indexing. In *Proceedings of the 5th International Conference on Recent Advances in Visual Information Systems (VISUAL 2002)*, volume 2314 of *Lecture Notes in Computer Science*, pages 36–49, Hsin Chu, Taiwan, Mar. 2002. Springer.

[29] J. T.-L. Wang, X. Wang, D. Shasha, and K. Zhang. MetricMap: an embedding technique for processing distance-based queries in metric spaces. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 35(5):973–987, 2005.

[30] R. Weber and K. Böhm. Trading quality for time with nearest-neighbor search. In *Proceedings of the 7th International Conference on Extending Database Technology (EDBT 2000)*, volume 1777 of *Lecture Notes in Computer Science*, pages 21–35, Konstanz, Germany, Mar. 2000. Springer.

[31] R. Weber, H.-J. Schek, and S. Blott. A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. In *Proceedings of the 24th International Conference on Very Large Data Bases (VLDB'98)*, pages 194–205, New York City, NY, Aug. 1998. Morgan Kaufmann.

[32] P. Zezula, P. Savino, G. Amato, and F. Rabitti. Approximate similarity retrieval with M-trees. *The VLDB Journal*, 7(4):275–293, 1998.