Approximate similarity search: A multi-faceted problem

Marco Patella ∗, Paolo Ciaccia

Università di Bologna - DEIS, viale Risorgimento, 2 - 40136, Bologna, Italy

Article info
Abstract

Article history:
Available online 26 September 2008

Keywords:
Similarity search
Approximate queries
Scheduling strategy

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 and that their performance is indeed the best among practical schedules.

© 2008 Elsevier B.V. All rights reserved.

1. Introduction

Similarity searching has been established as a fundamental paradigm for several modern applications [1]. Essentially, the problem is to find, in a set of objects, those which are more similar to a given query object. The similarity between any pair of objects is usually assessed by way of some distance function, being understood that low values of distance correspond to high degrees of similarity. The problem is as follows: Given a metric space \( M = (\Omega, d) \), where \( \Omega \) is a domain, also called the object space, and \( d : \Omega \times \Omega \to \mathbb{R}^+ \) 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 \( q \in \Omega \). Metric spaces are almost ubiquitous, and they include the \( D \)-dimensional vector space \( \mathbb{R}^D \) equipped with the Euclidean distance \( L_2 \) and the set \( \Sigma^* \) of (finite length) strings obtained from an alphabet of symbols \( \Sigma \) compared using 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. The most common types of similarity queries include (1) range queries, where all the objects in \( X \) whose distance to \( q \) does not exceed a user-specified threshold \( r \) are requested, and (2) \( k \)-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, mainly because the user is able to control the query selectivity, i.e., the cardinality of the result set, in the following we will concentrate on this kind of queries.\(^1\)

To speed up the resolution of similarity queries, a plethora of access structures have been proposed over the years: They can be broadly classified, depending on their field of applicability, as multi-dimensional (spatial) and metric access methods, where 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 [28,43]: 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.

In order to accelerate the search, the user is commonly offered a quality/time trade-off: For saving search time, she has to accept a degradation in the quality of the result, i.e., an error with respect to the exact case. The goal of approximate

\(^{1}\) We do not consider here the problem of searching in non-metric spaces [38].
similarity search is to reduce search times for similarity queries by possibly, but not necessarily, introducing an error in the result.

In this paper 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 and conclude.

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 \( k \) objects in the approximate result might not be the closest to 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 [36].
- For the same reason, the process of similarity search is typically iterative, because the user may be searching, using a feedback cycle [4], 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: Cost, typically 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 query \( q \) in \( X \) as \( \text{nn}_X(q) \) and the \( i \)th NN provided by an approximate algorithm as \( \tilde{\text{nn}}_X(q) \) (for simplicity, the algorithm is understood in the notation).

2. A classification schema

Based on the observation that exact similarity search is sometimes “difficult”, i.e., 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:

\[ \text{VS}_{L_p} \] (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.\(^2\) Specific classes can be obtained by instantiating \( p \); for instance, the class \( \text{VS}_{L_2} \)

\[ 1 \leq p < \infty, \quad L_p(x, y) = \left( \sum_{i=1}^{D} |x(i) - y(i)|^p \right)^{1/p}, \]

\[ L_\infty(x, y) = \max_{1 \leq i \leq D} (|x(i) - y(i)|). \]

\(^2\) We recall that the definition of the \( L_p \) distance between two points \( x \) and \( y \) in a \( D \)-dimensional space is as follows: \( L_p(x, y) = \left( \sum_{i=1}^{D} |x(i) - y(i)|^p \right)^{1/p} \).

---

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; for such cases, 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 [37].

**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 approximation techniques, assigning each of them to the proper class.

**Example 1.** Locality-Sensitive Hashing (LSH) [25] transforms a $D$-dimensional object $p$ into a sequence of $C$ bits (binary vector) $v(p)$. Since the $L_1$ distance between objects can be approximated by the Hamming (edit) distance between the corresponding binary vectors, LSH uses a 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 objects. Since approximation to the Hamming distance only yields for the $L_1$ metric, this technique is of class VS$_1$.

**Example 2.** Approximate nearest neighbor search techniques based on the VA-file [43] are presented in [42]. The VA-file is a sequential structure containing approximations of spatial objects 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 objects and only the $k$ closest ones are kept. The techniques in [42] 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 VS class.

**Example 3.** The P-Sphere tree [26] is a 2-level index structure for approximate 1-NN search. In order to find the nearest neighbor of query $q$, the leaf node closest to $q$ 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 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:

**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 those based on dimensionality reduction.

**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):

- **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.
- **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 CS class, since the approximate results are chosen by considering only their bit vector approximations.

**Example 5.** The BBD-tree [3] 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.\(^3\) To reduce the number of tree nodes accessed, during the search the

---

\(^3\) However, the dependency on the space dimensionality $D$ is exponential.
query radius is reduced by a factor of $\epsilon$ with respect to the radius used for exact search. Therefore, this method can be classified as $\text{RC}_{\text{AP}}$.

**Example 6.** Three different algorithms to solve approximate k-NN queries with M-tree [18] are presented in [45]. The first one reduces the current searching radius of the k-NN query by a factor of $\epsilon$, thus it applies aggressive pruning ($\text{RC}_{\text{AP}}$ 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 $\text{RC}_{\text{ES}}$ class).

**Example 7.** The technique proposed in [24] 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 considered dimensions. This technique, therefore, combines characteristics of both classes $\text{CS}$ (since only some dimensions are used) and $\text{RC}_{\text{ES}}$ (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.\(^4\) The classification we give is as follows:

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{NG}$</td>
<td>(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.</td>
<td></td>
</tr>
<tr>
<td>$\text{DG}$</td>
<td>(deterministic guarantees) Techniques in this class are able to deterministically bound from above the error introduced by approximation.</td>
<td></td>
</tr>
<tr>
<td>$\text{PG}$</td>
<td>(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 ($&lt; 100%$) 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 [33].</td>
<td></td>
</tr>
<tr>
<td>$\text{PG}_{\text{par}}$</td>
<td>(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.</td>
<td></td>
</tr>
<tr>
<td>$\text{PG}_{\text{par}}$</td>
<td>(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.</td>
<td></td>
</tr>
</tbody>
</table>

**Example 8.** The third technique proposed in [45] (see **Example 6**), where the search is stopped when the distance improvement falls below an user-specified threshold, is in the $\text{NG}$ class, because no guarantees can be given on the accuracy of the approximate result.

**Example 9.** The algorithm for approximate search proposed for BBD-trees in [3] (see also **Example 5**), and the first technique proposed in [45] both use a value $\epsilon$ to reduce the query radius during the search. In both cases, it is guaranteed that the error, measured as $d(q, \tilde{n}_{\text{hit}}(q))/d(q, n_{\text{hit}}(q)) - 1$, cannot exceed $\epsilon$, thus both techniques belong to class $\text{DG}$.

**Example 10.** DBIN is a 2-level index for solving the k-NN problem [5]. 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 $\text{PG}_{\text{par}}$ class.

**Example 11.** The PAC (Probably Approximately Correct) technique proposed in [16] 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

---

\(^4\) Quality evaluation in approximate similarity queries is discussed in detail in [17].
accuracy threshold \( \epsilon \) with a probability limited by the user-specified confidence value \( \delta \). To guarantee this, the distance between the query objects and its 1-NN is estimated from the distance distribution [19] 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 \( \text{PG}_{\text{ngpar}} \) 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 structures that are not bound at construction time and can be used with different parameters’ values, according to current user’s needs.

\( \text{SA} \) (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.

\( \text{IA} \) (interactive approach) Methods in this class are not bound to a specific value for 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, for example, by requesting a maximum error of 0%.

Example 12. In the P-Sphere technique presented in [26] (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 \( \text{SA} \) class.

Example 13. The generalized NN search proposed in [28] is an approach for high-dimensional 1-NN search. The key idea here is to find, at query time, a suitable projection to reduce the space dimensionality; then, the 1-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. Since the user can specify, at query time, the value of \( D' \), this method can be classified as \( \text{IA} \).

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.

\( \text{FastMap [23]}) \) (\( \text{MS, CS, NG, SA} \)). The FastMap technique [23] 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 \( \text{MS} \) class; the transformation of the space leads to a transformation of the distance used to compare objects, thus this technique is in the \( \text{CS} \) class; in the paper, the authors give no guarantee on the error introduced for distance in the target space, hence the quality guarantee class is \( \text{NG} \); 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 \( \text{SA} \) class.

\( \text{DBIN [5]} \) (\( \text{VS, RC}_{\text{ES}}, \text{PG}_{\text{ngpar}}, \text{IA} \)). The DBIN (density based indexing) method was presented in [5] 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 1-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 1-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 \( \text{VS} \); the search is early terminated by using data distribution, thus the class of this technique is \( \text{RC}_{\text{ES}} \); as for quality

---

5 The error between exact distances and distances on 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.
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 $\text{PG}_{\text{par}}$; since the user can specify the tolerance parameter, used to stop the search, this method is in class $\text{IA}$.

**PAC [16]:** $(\text{MS, RC, PG}_{\text{par}}, \text{IA})$. PAC (probably approximately correct) nearest neighbor queries, introduced in [16], represent a probabilistic approach to approximate 1-NN search in metric spaces, where the error in the result can exceed a specified accuracy threshold $\epsilon$ with a probability that is limited by a confidence parameter $\delta$. 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 $\text{MS}$ class). The only information that is needed by the algorithm to prune index nodes from the search is the value of $r_1(q)$, the maximum value of distance from the query object $q$ for which the probability that the exact 1-NN of $q$ has a distance lower than $r$ is not greater than $\delta$:

$$r_1(q) = \sup \{ r | \Pr\{d(q, n_{\text{hit}}(q)) \leq r \} \leq \delta \} \quad (1)$$

In [16], this value is estimated by using the distance distribution of indexed objects with respect to the query object, which is obtained through sampling and stored as an histogram; it is therefore clear that this approach is probabilistic and non-parametric ($\text{PG}_{\text{par}}$ 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 $r_1(q)$ and of the $\epsilon$ parameter, thus the class of this approach is $\text{RC}$ (actually, both $\text{RC}_{\text{AP}}$ and $\text{RC}_{\text{ES}}$); finally, since the accuracy and the confidence parameters ($\epsilon$ and $\delta$, respectively) can be specified at query time, this technique belongs to the $\text{IA}$ class.

**VA-BND [42]:** $(\text{VS, RC}_{\text{AP}}, \text{PG}_{\text{par}}, \text{IA})$. Two approximate query evaluation techniques are presented in [42] for the VA-File. The VA-File structure [43] approximates $D$-dimensional points 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 objects corresponding to approximations included in the result of the previous scan, so-called “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, albeit computation of bounds can be a difficult task if complex metrics are used, all approximate techniques developed for this structure fall in the $\text{VS}$ 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 $\alpha$ to adapt computed bounds: Higher values of $\alpha$ 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 $\text{RC}_{\text{AP}}$. 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 $\alpha$; therefore, the class for this method is $\text{PG}_{\text{gapar}}$. Finally, since the parameter $\alpha$ can be specified at query time, the VA-BND technique is in class $\text{IA}$.

$\text{VA-LOW}$ [42]: $\{\text{VS}, \text{CS}, \text{DG}, \text{SA}\}$. The second approximate technique for the VA-file (also presented in [42]) 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 objects, this method can be classified as $\text{CS}$. 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 $\text{DG}$ class. As for the interaction with the user, it is clear that the only parameter used, i.e., the number of bits used for objects’ approximation, has to be specified before the actual VA-File is built, so that the class for this technique is $\text{SA}$.

$\text{Probabilistic Proximity Search}$ [15]: $\{\text{MS}, \text{RC}_{\text{AP}}, \text{PG}_{\text{gapar}}, \text{IA}\}$. The technique described in [14,15] is basically an adaptation of search radius reduction to pivot-based searching algorithms. 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: $\text{MS}$ (because pivot-based algorithms are applicable to generic metric spaces), $\text{RC}_{\text{AP}}$ (aggressive pruning is used by reducing the search radius), $\text{PG}_{\text{gapar}}$ (guarantees on the result quality are probabilistic and non-parametric), and $\text{IA}$ (a regular pivot-based index is used and the confidence level can be expressed by the user at query time).

$\text{Approximate/On-Line NN Queries}$ [32]: $\{\text{VS/MS, RC}_{\text{AP}}, \text{NG, IA}\}$. In [32], several techniques are presented to perform approximate searches in high-dimensional spaces using $\text{R}$-tree-like structures [27]. Basically, recognizing the fact that the difficult task in $k$-NN searching is to guarantee the correctness of the result, the authors propose several heuristics for aggressively pruning index nodes. Two of such heuristics use Monte Carlo simulation to estimate the probability of finding a better result in a given node, while the remaining three strategies estimate such probability by using distance bounds for the hyper-rectangular region associated to each node. Nodes leading to a probability lower than a user-specified threshold $\alpha$ are pruned from the search. Although the latter techniques can be in principle extended to other metric indices, like the M-tree and are thus in class $\text{MS}$, the ones based on Monte Carlo simulation are restricted to vector spaces, falling in the $\text{VS}$ class. For the other coordinates, the classification is as follows: $\text{RC}_{\text{AP}}$, $\text{NG}$ (no guarantees can be given on the quality of the result), and $\text{IA}$ (since the user can control the approximation quality by choosing the $\alpha$ parameter).

$\text{Proximity Searching with Order Permutations}$ [13]: $\{\text{MS}, \text{RC}_{\text{ES}}, \text{NG, IA}\}$. The algorithm presented in [12,13] is a modification of the basic pivot-based LAESA sequential algorithm [34] to avoid a complete analysis of the table of stored distances. To this end, for each point $p$ in the data set the pivots are sorted from nearest to farthest; the same is performed for the query. Then, points in the data set are sorted for increasing similarity of their sorted lists of pivots to the one of $q$, e.g., by using the Spearman’s Footrule distance [22]. Finally, only a user-specified fraction of points is examined (again, although only the algorithm for solving range queries is given, it can be easily extended to $k$-NN queries). The classification of this technique is therefore: $\text{MS}$, $\text{RC}_{\text{ES}}$, $\text{NG}$ (since no guarantees on result can be provided), and $\text{IA}$.

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 $\text{RC}$, $\text{PG}_{\text{gapar}}$, and $\text{IA}$ 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 ($\epsilon$ and $\alpha$, 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.

---

6 Although only the range search algorithm is given by the authors, the technique can be easily extended to k-NN queries.
In the same way, one could discover that similarities exist between these two approaches and Probabilistic Proximity Search. From Table 1 we also see that techniques based on radius shrinking [3,21,45], where the distance to the current k-th NN is reduced by a factor ϵ in order to increase the probability of pruning objects, are classified as RC_EP (since a higher number of objects can be pruned), DG (the error is guaranteed to be not higher than ϵ), and IA (the value of ϵ can be specified by the user at query time). Such techniques are based on a same principle and differences in performance are only imputable to the underlying index structure.

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 [16]. Finally, it is clear that interactive approaches are more practical 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 costs needed to obtain the 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 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 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.

We consider an 1-NN search over a flat index that organizes objects in X as a collection \( N \) of n nodes, since this is the case more amenable to be formally characterized, and leave out for future investigation the general cases of trees with arbitrary height and of k-NN search with \( k > 1 \). We assume that, for each node \( N_i \), the index provides a compact representation of (the actual content of) \( N_i \), which is used to determine bounds on the distance between the query and the objects in \( N_i \). For instance, the representation provided by a ball-partitioning index, like the List of Clusters [9], consists of a center \( c_i \) and a radius \( r_i \), that together determine the “extension” of node \( N_i \) in the metric space. Pivot-based matrices also fit this scenario, by considering that each node \( N_i \) stores a single object \( p_i \) and that the representation of \( N_i \) consists of the distances between \( p_i \) and the pivots.

On-line 1-NN search in the above-described scenario can be performed as illustrated in Algorithm 1. All nodes are initially inserted in a priority queue \( PQ \) and they are accessed one-by-one according to a scheduling policy \( \Pi \) (line 2) until the stopping condition is met or \( PQ \) gets empty (line 1). Every node extracted from the queue is first checked to see whether it can contain a better solution (line 3); only in this case, the node is accessed and its objects compared to the query, possibly leading to improve the current solution (line 6).

Each schedule \( \Pi \) 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 \( \Pi_i \) is the leaf that schedule \( \Pi \) will fetch at step \( i \). In order to build \( \Pi \), information about the query \( q \) and the compact representation of each node \( N_i \) provided by the index are used: For example, the MinDist policy [6,29] orders nodes for increasing values of the lower bound of the distance between \( q \) and any object in \( N_i \) (MinDist\( (q, N_i) = \max[d(q, c_i) - r_i, 0] \))

```
Algorithm 1. On-line 1-NN.
```

```plaintext
Input: query q, collection of nodes N = {N1, ..., Nn}, scheduling policy \( \Pi \)
State: current approximate result \( \tilde{m}(q) \) and priority queue \( PQ \)
Output: updated approximate 1-NN of q in X, \( \tilde{m}(q) \)
1: \( while (\neg(PQ = \emptyset) \vee \text{Stop}(q, \tilde{m}(q))) \) do
2: \( N_i \leftarrow \text{DEQUEUE}(PQ, \Pi) \) \quad \text{Extract the first entry from PQ respecting the scheduling policy \( \Pi \)}
3: \( if \neg\text{PRUNE}(N_i, \tilde{m}(q)) \) then
4: \( for \ all \ points \ p \ in N_i \) do
5: \( if d(q, p) < d(q, \tilde{m}(q)) \) then
6: \( \tilde{m}(q) \\leftarrow p \quad \text{\( p \) is the new 1-NN \)
7: return \( \tilde{m}(q) \)
```
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 representation of $N_i$ provided by the index, 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.7

Given the above, it is 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 show how it is possible to obtain the best results by minimizing the number of accessed nodes. Although focusing only on node accesses ignores other factors that can influence performance (e.g., overhead of managing the priority queue and distance computations), it has the clear advantage of focusing on the role of scheduling, which is what we aim to understand here.

Further, if the number of objects in each node is almost constant, the number of accessed nodes is also tightly correlated with the number of computed distances.

To start with, we define what an optimal schedule for approximate search is.

**Definition 1 (Optimal Schedule for Query $q$).** Let $N$ be a collection of nodes partitioning a data set $X \subset Ω$, and $q \in Ω$ a query point.

We say that schedule $Π$ cost-dominates at distance level $θ$ schedule $Π'$ iff $\text{Cost}(q; Π, θ) < \text{Cost}(q; Π', θ)$, where $\text{Cost}(q; Π, θ)$ measures the (minimum) cost paid for query $q$ to return a result with distance $\leq θ$ 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 $d(q; Π, c) < d(q; Π', c)$, where $d(q; Π, c)$ 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 Fig. 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 level $c_2$.

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

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

**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 $c' = \text{Cost}(q; Π', θ') < c = \text{Cost}(q; Π, θ')$. 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. □

---

7 This happens with the scheduling proposed in [2], 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).
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 concerning the possibility of deriving an optimal schedule for arbitrary query points and metric spaces.

**Lemma 2.** Let \( \mathcal{N} \) be a collection of \( n > 1 \) nodes partitioning a data set \( X \subset \Omega \), and \( q \in \Omega \) be a query point. If there exists at least one node in \( \mathcal{N} \) that (1) cannot be pruned by using the information provided by the nodes’ representations and (2) does not contain a nearest neighbor for \( q \), then there exists no optimal schedule for \( q \).

**Proof.** First observe that the definition of optimality implies that a schedule \( \Pi \) is optimal iff the first accessed node contains a nearest neighbor of \( q \), otherwise there would be another (better) schedule that after one access dominates \( \Pi \). Without loss of generality, assume that exactly two nodes in the queue, say \( N_1 \) and \( N_2 \), cannot be pruned. According to the lemma’s hypothesis, only one of them contains a nearest neighbor of \( q \). Assume \( \Pi \) is an optimal schedule for \( q \) and, without loss of generality, that \( \Pi \) first fetches node \( N_1 \). Since \( N_2 \) has not been pruned, it might be the case that it provides a better result than \( N_1 \), which in turn implies that \( \Pi \) is not optimal. \( \square \)

Above lemma shows that single-query optimality is impossible to attain for approximate similarity search. This is a direct consequence of the limited information provided by the index and of the very definition of optimality.

The first point highlights the, indeed trivial, observation that the same index representation is, in the general case, compatible with a multitude of data sets, thus the node(s) containing a nearest neighbor of query \( q \) cannot be exactly determined in advance. As a simple example, consider the case of two data sets, \( X_A \) and \( X_B \), leading respectively to collections \( \mathcal{N}_A \) and \( \mathcal{N}_B \), each having just two nodes \( (N_{A1} \ and N_{A2}, N_{B1} \ and N_{B2}, respectively) \). The 1-NN of \( q \) is in \( N_{A1} \) for \( X_A \) and in \( N_{B2} \) for \( X_B \) (see Fig. 2). If the description of \( N_{A1} \) equals that of \( N_{B1} \) and the description of \( N_{A2} \) equals that of \( N_{B2} \), the query \( q \) in both cases would “see” the same nodes. Then, any scheduling policy will act the same on both indices, which is enough to see that no schedule can be optimal on both data sets.

It is also interesting to observe that the absence of an optimal schedule holds even for the 0-error case, in spite of the I/O-optimality (proved in [6,29]) of the MinDist policy. Indeed, although MinDist minimizes the number of I/Os of any provable exact search, this does not imply that it is the schedule that can return earlier the exact result in an on-line search (this is also demonstrated by results in the following). As also observed in [16], the hard problem in exact NN search is not locating the nearest neighbor, but reaching the condition that allows the search to be stopped because correctness of the result can be guaranteed.

**Lemma 2** can also be interpreted as the impossibility of implementing an optimal on-line NN search, in which at each step one gets the best possible result up to that point and eventually obtains the correct nearest neighbor. However, as argued above, this would imply that the first approximate 1-NN coincides with the exact one, which is impossible for arbitrary queries.

**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 single query, are optimal-on-the-average. A schedule \( \Pi \) that is either cost- or distance-optimal-on-the-average has the property that no other schedule \( \Pi' \) 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 nodes yet to be accessed, maximizes the quantities listed in Table 2, where \( G_i(r) \) is the distance distribution of the 1-NN of \( q \) in node \( N_i \), \( G_i(r) = \Pr[d(q, nn_{N_i}^1(q)) \leq r] \), and \( d^+ \) denotes the maximum distance value for \( d \) over \( M \).

---

8 Note that here we are considering the general case where multiple points can share the minimum distance from \( q \).

9 Arguments used in the proof of the lemma can also be applied to show that neither cost-optimality nor distance-optimality can be obtained for any distance/cost level.
Table 2

<table>
<thead>
<tr>
<th>Type of optimality</th>
<th>Scenario</th>
<th>Quantity to maximize at each step</th>
</tr>
</thead>
<tbody>
<tr>
<td>cost-optimal</td>
<td>given θ, minimize average cost</td>
<td>( G_i(θ) )</td>
</tr>
<tr>
<td>distance-optimal</td>
<td>given c, minimize average distance</td>
<td>( \int_0^{d^+} G_i(r) , dr )</td>
</tr>
</tbody>
</table>

**Proof.** We first prove the result for cost-optimal schedules. To this end, we should first estimate the number of nodes accessed before Algorithm 1 stops, i.e., to compute the probability, \( p_{\text{stop}}(c, θ; Π) \), that the search algorithm, using schedule \( Π \), will find in no more than \( c \) steps a point whose distance from the query \( q \) is not higher than \( θ \):

\[
p_{\text{stop}}(c, θ; Π) = \Pr\left( \min_{i \leq c} d(q, mn_{N_i}(q)) \leq θ \right) = 1 - \prod_{i=1}^{c} \Pr\left( d(q, mn_{N_i}(q)) > θ \right) = 1 - \prod_{i=1}^{c} (1 - G_{π_i}(θ))
\]

i.e., \( p_{\text{stop}}(c, θ; Π) \) equals the probability to find, in any of the first \( c \) nodes of \( Π \), a point whose distance to the query is not higher than \( θ \).

From Eq. (2), it follows that \( Π \) is cost-optimal-on-the-average if:

\[
p_{\text{stop}}(c, θ; Π) \geq p_{\text{stop}}(c, θ; Π') \quad \forall Π', \forall c \in [1, M - 1]
\]

which amounts to choose, at each step \( j \), the node that maximizes the (overall) probability of stopping the search among non-fetched nodes, i.e., \( Π_j = \text{arg max}_i \{ G_i(θ) \} \).

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 [19]:

\[
E[d(q, mn_{N_i}(q))] = d^+ - \int_0^{d^+} G_i(r) \, dr
\]

from which the result immediately follows. □

We tested the performance of optimal-on-the-average schedules on several real data sets. Fig. 3 reports results for the Corel data set, containing 68,040 32-dimensional color histograms extracted from a collection of Corel images and compared using the Euclidean distance.\(^{10}\) The index structure was built using the 4000+ leaf nodes of an M-tree. For the distance-optimal schedule that we use in the experiments, the probability \( G_i(θ) \) of node \( N_i \) should be estimated for each query. To this end, in order to keep a low memory overhead, for each node \( N_i \) we divided (a sample of) the queries in 10 classes, according to their distance to \( N_i \), and computed the average value of \( \int_0^{d^+} G_i(r) \, dr \) for each class. At query time, the specific value for each node \( N_i \) is chosen by looking at the class of the query at hand.

In the first experiment we contrast the performance of the distance-optimal schedule with that of MinDist, which is the optimal schedule for exact k-NN search, and with that of a Random schedule, which at each step selects a random yet-unaccessed node. Fig. 3(a) shows the distance of the current 1-NN after \( c \) nodes have been accessed, averaged over 682 queries extracted from the data set.

These results (similar ones were obtained for all the other tested datasets) lead us to draw several important conclusions:

- The distance-optimal schedule is indeed able to obtain very good results in the earlier stages of the search: After accessing a single node, this schedule finds an object which is only 27% farther than the exact 1-NN.
- Neither the MinDist nor the Random schedules are able to find good results with just a few node accesses. As it can be expected, the former strategy is clearly better than the latter. After 10 nodes have been accessed, the distance-optimal schedule is only 6% away from the correct 1-NN, while MinDist and Random are 96% and 375% away, respectively.
- The graphs also show the total average cost for solving an exact 1-NN search. In this case, the winner is clearly the MinDist strategy which, as said, is provably optimal; still, the cost of the distance-optimal strategy is only 0.05% higher. The Random policy performs surprisingly well, with only a 6% overhead over MinDist.
- The behavior of the distance-optimal schedule is particularly good if one considers the minimal overhead required: Only 10 real values, i.e., the actual value of Eq. (4) for each query class, are stored for each node.

Our second experiment considers two scheduling heuristics that have been proposed in [9] for the List of Clusters. The MinMaxDist scheduling orders nodes by increasing values of MinMaxDist\((q, N_i) = d(q, c_i)\), whereas the MaxDist policy considers for each node the quantity MaxDist\((q, N_i) = d(q, c_i) + r_i\). Results shown in Fig. 3(b) demonstrate that such heuristics

\(^{10}\) This data set is available at the UCI Knowledge Discovery site: http://kdd.ics.uci.edu.
Fig. 3. Result quality as a function of the number of accessed nodes for different scheduling policies: (a) Random, MinDist, and distance-optimal, (b) MinMaxDist, MaxDist, and distance-optimal.

perform much better than MinDist (this was also observed in [9]) and that they quite well approach the performance of the distance-optimal schedule. Since such schedules, unlike the distance-optimal one, do not need any additional information, it would be interesting to go beyond purely empirical results and to study under which conditions their good relative performance levels can be theoretically guaranteed. Further, investigating the trade-off between the amount of pre-computed information stored for each node and the performance of optimal-on-the-average schedules is an important research issue, since they are provably the best practical schedules for on-line 1-NN queries.

To conclude, we briefly 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 [6,29]: If we take the optimistic case, where all the mass probability $G_i(q)$ 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 same light, MinMaxDist implements the pessimistic case, since MinMaxDist$(q, N_i)$ is an upper bound on the distance between $q$ and objects in $N_i$. Finally, in the DBIN approach [5], 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).

5. Conclusions

In this paper we have introduced a schema for classifying approaches to the approximate similarity search problem and have shown how several 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 another, often underestimated, facet of the approximate search problem, namely the scheduling of index nodes: Even if limited to a specific class of approximation techniques, our formal and preliminary experimental result on optimality of schedules can be viewed as a starting point upon which more general theories could be built in the future.

Acknowledgements

The authors thank Dr. Alessandro Linari for helping with the experiments on optimal-on-the-average schedules.

References


[38] T. Skopal, Unified framework for fast and exact approximate search in dissimilarity spaces, ACM Transactions on Database Systems 32 (4).

