A Comparison of Pivot Selection Techniques for Permutation-Based Indexing

Giuseppe Amato, Andrea Esuli, Fabrizio Falchi

Istituto di Scienza e Tecnologie dell'Informazione "A. Faedo",
via G. Moruzzi 1, Pisa 56124, Italy
{*firstname*}.{*lastname*}*@isti.cnr.it*

Abstract

Recently, permutation based indexes have attracted interest in the area of similarity search. The basic idea of permutation based indexes is that data objects are represented as appropriately generated permutations of a set of pivots (or reference objects). Similarity queries are executed by searching for data objects whose permutation representation is similar to that of the query, following the assumption that similar objects are represented by similar permutations of the pivots.

In the context of permutation-based indexing, most authors propose to select pivots randomly from the data set, given that traditional pivot selection techniques do not reveal better performance. However, to the best of our knowledge, no rigorous comparison has been performed yet. In this paper we compare five pivot selection techniques on three permutation-based similarity access methods. Among those, we propose a novel technique specifically designed for permutations. Two significant observations emerge from our tests. First, random selection is always outperformed by at least one of the tested techniques. Second, there is not a technique that is universally the best for all permutation-based access methods; rather different techniques are optimal for different methods. This indicates that the pivot selection technique should be considered as an integrating and relevant part of any permutation-based access method.

Keywords: permutation-based, pivot, metric space, similarity search, inverted files, content based image retrieval

1. Introduction

Given a set of objects C from a domain $\mathcal{D}$, a *distance function* $d : \mathcal{D} \times \mathcal{D} \to \mathbb{R}$, and a query object $q \in \mathcal{D}$, a similarity search problem can be generally defined as the problem of finding a subset $S \subset C$ of the objects that are closer to q with respect to d. Specific formulations of the problem can, for example, require to find the k closest objects (k-nearest neighbors search, k-NN), i.e., $|S| = k$ and $\forall x \in S, y \in (C \setminus S)$ $(d(x, q) \leq d(y, q))$, or all the objects that are closer than a given threshold distance t. i.e., $S = \{x | x \in C \wedge d(x, q) \leq t\}$. The k-NN formulation is the most common one.

Similarity search is a difficult problem and various indexing schema have been defined to process similarity queries efficiently. Good surveys of the various approaches proposed in literature can be found in [39, 34]. However, in most applications, as for instance multimedia retrieval, an exact solution to the similarity search problem is not strictly required. In these cases, performing an approximate similarity search [40, 30] is sufficient. Accepting even a small degree of approximation in results allows to obtain them much more efficiently.

Permutation-based indexes have been proposed as a new approach to efficient and effective approximate similarity search [2, 12, 16, 28]. In permutation-based indexes, data objects and queries are represented as appropriate permutations of a set of n pivots $P = \{p_1 \dots p_n\} \subset \mathcal{D}$. Formally, every object $o \in \mathcal{D}$ is associated with a permutation Π_o that lists the identifiers of the pivots by their closeness to o, i.e., $\forall j \in \{1, 2, \dots, n-1\},\ d(o, p_{\Pi_o(j)}) \leq d(o, p_{\Pi_o(j+1)})$, where $p_{\Pi_o(j)}$ indicates the pivot at position j in the permutation associated with object o. For convenience, we denote the position of a pivot p_i, in the permutation of an object $o \in \mathcal{D}$, as $\Pi_o^{-1}(i)$ so that $\Pi_o(\Pi_o^{-1}(i)) = i$.

The similarity between objects is approximated by comparing their representation in terms of permutations. The basic intuition is that if the permutations relative to two objects are similar, i.e. the two objects *see* the pivots in a similar order of distance, then the two objects are likely to be similar also with respect to the original distance function d.

Once the set of pivots P is defined it must be kept fixed for all the indexed objects and queries, because the permutations deriving from different sets of pivots are not comparable. A selection of a "good" set of pivots is thus an important step in the indexing process, where the "goodness" of the set is

This is a revised and extended version of a paper appeared as [1].

measured by the effectiveness and efficacy of the resulting index structure at search time.

Permutation based methods share some ideas with the Shared Nearest Neighbors methods (SNN) [22, 32, 14]. These methods introduce the concept of secondary similarity measures, which evaluates the similarity among two objects by considering the amount of overlap of their neighborhoods. The neighborhood of an object is determined using the original distance and all objects of the dataset. Secondary similarity measures have been shown to be able to reduce the impact of the curse of dimensionality in cases in which the discriminative power of the primary similarity measure is reduced by the high dimensionality of the similarity space. The difference between the permutation-based methods and the SNN methods is that permutation-based methods encode original objects with neighbor objects taken from a very small subset of the entire dataset, rather than the entire dataset. Moreover, the primary purpose of this encoding is to build efficient and scalable approximate similarity search index structures, rather than computing a better distance than the original distance, as SNN method do. In addition, no pivot selection technique is needed by SNN methods, given that they use the entire dataset to determine the neighborhood of objects.

In the field of permutation-based access methods the most commonly adopted technique for the definition of P is to randomly select the n objects from C [2, 12, 16]. Even though there is a relatively rich literature on pivot selection techniques for the general class of *pivot-based* access methods [39] (see Sections 2 and 3), to the best of our knowledge, no rigorous comparison of the effectiveness of the various selection techniques, when used in combination with permutation-based access methods, has been performed yet. In this paper we compare five techniques for the definition of sets of pivots to be used by permutation-based access methods. One of the techniques that we compare is a novel proposal that we have designed to be used with permutation based index.

In summary, the contribution of this paper is twofold.

1. We test various pivot selection techniques, including random selection, on a number of permutation based indexes. An interesting result is that different selection methods were optimal for different index schema. In fact, the way in which permutations are used, by different indexing schema, is basically different, and this is reflected in the pivot selection techniques.

2. We propose and compare a new pivot selection criterium, expressly designed to be used with permutation-based indexes. This method is clearly superior when used with the MI-File index [2].

The paper is structured as follows. In Section 2 we discuss related work. Section 3 presents the techniques being compared. The tested similarity search access methods are presented in Section 4. Section 5 describes the experiments and comments their results. Conclusion and future work are given in Section 6.

2. Related Work

The study of pivot selection techniques for access methods usually classified as pivot-based [39] has been an active research topic, in the field of similarity search in metric spaces, since the nineties. Most access methods make use of pivots to reduce the number of data objects accessed during similarity query execution. The choice of the pivots plays a relevant role in allowing the access methods to achieve their best performance. In an early work by Shapiro [37], it was noticed that good performance were obtained by locating pivots *far away* from data clusters. In [8, 26, 38], following this intuition, several heuristics were proposed to select pivots between the *outliers* and far away from each other.

In [18] it is shown that it is possible to find an optimal set of pivots selecting them as the vertices of a large regular simplex containing all the objects of the database but this result does not apply to general metric spaces.

Pivot selection techniques that maximize a lower bound estimate of the original distance d by means of a *pivoted distance* were exploited in [11] (see Section 3.3). For these techniques is was also observed that while good pivots are usually outliers, not all outliers can be good pivots [9]. In [10, 31], the problem of dynamic pivot selection as the database grows is faced. In [25] Principal Component Analysis (PCA) has been proposed for pivot selection. Principal components (PC) of the dataset are identified by applying PCA on it (actually a subset to make the method computationally feasible) and the objects in the dataset that are best aligned with PC vectors are selected as pivots. In [38] it was proposed to select the corners of the data set as pivots for Vantage Point Tree (VPT). Multi-Vantage Point Tree (MVPT) [7] selects multiple corners.

Works that use permutation-based indexing techniques have mostly performed a random selection of pivots [2, 16, 12] following the observation that

the role of pivots in permutation-based indexes appears to be substantially different from the one they have in traditional pivot-based access methods. In fact, the use of previous selection techniques, with permutation based indexes, did not reveal significant advantages. At the best of our knowledge, the only report on the definition of a specific selection techniques for permutation-based indexing is in [12], where it was mentioned that no significant improvement, with respect to random selection, was obtained by maximizing or minimizing the Spearman Rho distance through a greedy algorithm.

3. Pivot Selection Techniques

Permutation-based access methods use pivots to build permutations that represent data objects. However, different permutation-based indexes make different use of the permutations. We can broadly identify two different roles played by the permutations in these indexes schema:

1. Rank data objects according to the distance (or dissimilarity) between permutations, rather than the original distance.
2. Provide focused access in the database to identify and retrieve candidate objects, which the similarity search result is obtained from.

Accordingly, choosing a good set of pivots to build permutations, may have two different objectives:

1. Produce a distance between permutations that ranks objects with a good approximation, (w.r.t. ranks obtained with the original distance).
2. Help to identify good set of candidate objects to speed-up search process and obtain good precision, at the same time.

This paper compares four promising selection techniques used in combination with different permutation-based indexes, in order to make a comprehensive evaluation and also to identify the specific features that can be exploited in the various cases.

Both indexes and selections techniques are discussed in the following. However, we anticipate here their classification following this scheme. Regarding the tested access methods, in Permutations Spearman Rho (Section 4.1) pivots are used only to approximate the original distance, instead PP-Index (Section 4.3) only uses pivots for focused accessing the database while MI-File (Section 4.2) employs both approaches.

The objective of the tested pivots selections techniques are more difficult to understand. However, we expect clustering approaches (k-medoids, Section 3.2) to be useful for identifying good set of candidates, while minimizing the largest distance of an object in the dataset from the closest pivot (Farthest-First Traversal, Section 3.1) should be more appropriate for approximating the original distance. When both objectives have to be taken into account, we believe that our proposed approach (Balancing Pivot-Position, Section 3.4) should be appropriate because it has the goal of balancing the distribution of the pivots in the various positions of the permutations. Finally, the well known Pivoted Space Incremental Selection (PSIS) (Section 3.3) have been proven to be effective for lower bounding the original distance by using the pivoted distance that relies on the triangle inequality of the metric space. This is not the case of permutation-based methods. Thus, we expect PSIS to be not an appropriate technique for any permutation-based method. As the baseline we tested the random technique, which samples pivots from the dataset following a uniform probability distribution.

The results reported in this paper are consistent with the theoretical considerations made in this section.

3.1. Farthest-First Traversal (FFT)

A very well known topic in metric spaces is the *k-center* NP-hard problem that asks: given a set of objects C and an integer k, find a subset P of k objects in C that minimizes the largest distance of any object in C from its closest object in P. FFT (so called by Dasgupta in [13]) finds a solution close to optimal by selecting an arbitrary object $p_1 \in C$ and choosing, at each subsequent iteration, the object $p_i \in C$ for which $\min_{1 \leq j < i} d(p_i, p_j)$ is maximum. FFT, in each iteration picks the farthest object from the current set of pivots.

In [21], it has been proven that FFT achieves an approximation, with respect to the optimal solution, of at most a factor of 2. Note that FFT actually tries to maximize the minimum distance between the pivots, which intuitively could be a desirable property of the resulting pivot set. In fact, in this way, selected pivots somehow represent different areas of the data space. This might help in obtaining a distance between permutations that exhibit a behavior similar to that of the original distance.

The computational cost of this algorithm is $O(n|C|)$ [21], where n is the number of requested pivots.

We expect this approach to result in good approximation of the original distance by using the similarity permutations methods such as Spearman

Rho. In fact, maximizing the minimum distance between the pivots should result in less noisy changes in the positions of each pivot in the permutations.

3.2. k-medoids (kMED)

Originally proposed in [23], k-medoids is a partitional clustering algorithm that tries to minimize the average distance between objects and selected cluster medoids. k-medoids is very similar to k-means. The difference is that it uses objects from the dataset as representatives of the centers of the clusters rather than computing centroids, which could be not possible in general metric spaces. Moreover, k-medoids is also more robust to noise and outliers because it minimizes the distances instead of their square. While FFT minimizes the largest distance of an object from its closest pivot, k-medoids minimizes the average distance of the objects from their closest pivot.

With this technique, the role of the selected pivots is that of representing dense area of the portion of the space where they are located. Intuitively, indexes that use permutations to identify promising sets of candidate objects might benefit from this method.

3.3. Pivoted Space Incremental Selection (PSIS)

In [11], a technique to select a set of pivots P, such that the *pivoted distance*, defined as $D_P(x, y) = \max_{p \in P} |d(x, p) - d(y, p)|$, is as close as possible to the original distance d, was proposed proposed. Given that D_P is a lower bound of the original distance d, the objective here is to select the pivots P so that the pivoted distance is, on average, maximized.

The author observed that the chosen pivots are outliers. However they also noticed that not all outliers are good pivots for maximizing the average D_P. The overall best between the methods they proposed is the incremental selection technique. This technique greedily selects the first and subsequent pivots maximizing D_P on a set of pairs of objects in C.

3.4. Balancing Pivot-Position occurrences (BPP)

While the other pivot selection techniques mainly originate from the literature on similarity search in metric spaces and clustering, in this section we propose a new algorithm specifically intended for permutation-based access methods.

This algorithm originates from the intuition that every pivot should have a relevant role in determining the positional properties of the indexed objects.

To clarify this idea, suppose that a pivots always appears in the same position in all the permutations associated with objects in the database. Such pivot is indeed useless, as it does not bring any information into the permutation. A similar case is when a pivot never appears in some positions of the permutations, as the information the pivot can bring into the permutation is less than the maximum possible.

The benefit of this balancing should be noted in both the quality of the approximation of the original distance (when permutations similarity measures are used such as the Spearman Rho) and the effective and efficiency in selecting a small portion of the dataset relying on the permutations.

The above intuition suggests that the distribution of the various pivots, in the various positions of the permutations, should be made uniform as much as possible. Let $c(p_i, j) = |\{\Pi_o : \Pi_o^{-1}(i) = j\}|$ be the number of permutations where p_i appears in position j. BPP tries to minimize the standard deviation of $c(p_i, j),\ 1 \leq j \leq n,\ 1 \leq i \leq |C|$ from the mean. Note that the mean value of $c(p_i, j),\ 1 \leq j \leq n$ is independent of the specific set of pivots and is always equal to $|C|/n$.

The algorithm starts by randomly selecting a set $P \in C$ of $\hat{n} > n$ candidate pivots and evaluating the permutations for all the objects $o \in C$ (or a subset $S \subset C$). At each iteration, the algorithm evaluates the effect of removing each $p_i \in P$ (or a fixed number t of candidate pivots) on the distribution of $c(p_i, j)$ and removes the pivot for which the minimum average standard deviation is obtained. The algorithm ends when the number of candidate pivots satisfies the request, i.e. $|P| = n$.

In [2] it was observed that the pivots that appear at the beginning of the permutations, i.e. the nearest pivots to the object, are the more relevant. Thus, in our experiments, we applied this general algorithm considering $c(p_i, j)$ for $1 \leq j \leq l$ where l is the actual length of the permutation we are considering. In this way, the pivots never or rarely appearing in the first l positions are discarded.

The complexity of the algorithm is thus $O(\hat{n}|S|)$ for initialization using the distance d, and $O(t\hat{n}^2|S|)$ for the iterative selection where the cost is the evaluation of each candidate pivot occurrence in the permutations.

The BPP algorithm tries to overcome a phenomenon similar to the *hubness* already studied other contexts [33, 35, 19, 36]. Hubness is perfectly defined by Schnitzer et al. in [36] as *"a recently discovered general problem of machine learning in high dimensional data spaces. Hub objects have a small distance to an exceptionally large number of data points, and anti-hubs*

are far from all other data points". Hub objects tend to appear in the result set of several queries; anti-hubs almost never appear as result of any query. Following the consideration made while formulating BPP, it is evident that hubs and anti-hubs are not good candidates as pivots. Moreover, BPP has even stricter requirements, as it searches for a set of pivots equally distributed across the various positions of the permutations. In fact, in order to obtain this, it is not sufficient to avoid/selecting hub/anti-hub objects. For instance, it might happen that an object that is not an hub, considering the entire dataset, might appear always in the first positions of the permutations, if the pivots are not chosen carefully. This might happen when all other chosen pivots are all slightly farther than the hub pivot from most of the objects of the dataset. The quality of a set of pivots, with respect to a dataset, have to be judged considering the entire set of pivots together, rather than evaluating each pivot against the dataset independently of the other pivots.

4. Permutation-Based Similarity Access Methods

We have compared the pivot selection techniques on three permutation based index structures that reasonably cover the various approaches adopted in literature by access methods based on permutations.

4.1. Permutations Spearman Rho (PSR)

The idea of predicting the closeness between elements comparing the way they "see" a set of pivots was originally proposed in [12]. As distance between permutations, Spearman Rho, Kendall Tau and Spearman Footrule [17] were tested. Spearman Rho revealed better performance. Given two permutations Π_{o_1} and Π_{o_2}, Spearman Rho is defined as:

$$S_\rho(\Pi_{o_1}, \Pi_{o_2}) = \sqrt{\sum_{1 \leq i \leq n} (\Pi_{o_1}^{-1}(i) - \Pi_{o_2}^{-1}(i))^2}$$

When a k-NN search is performed, a candidate set of results of size $k' > k$ is retrieved considering the similarity of the permutations based on the distance S_ρ (in our experiments we fixed $k' = 10k$). This set is then reordered considering the original distance d. In [12] an optimal incremental sorting [29] was used to improve efficiency, when the candidate set of results to be retrieved using the Spearman Rho is not known in advance. In this work

we just perform a linear scan of the permutations defining the size of the candidate set in advance.

As already mentioned and stated in other works [2, 16], the most relevant information of the permutation Π_o lies in the pivots at the beginning of the permutation. Accordingly, in addition to the full permutations, tests were also executed considering permutation prefixes of length l. To compare these truncated permutations, we used the Spearman Rho distance with location parameter $S_{\rho,l}$ defined in [17], which is intended for the comparison of top-l lists. The definition of $S_{\rho,l}$ is very similar to the definition of S_ρ given above. The difference is that the position of the pivots occurring out of the truncated permutation is considered to be $l+1$. More formally, in place of $\Pi_o^{-1}(i)$ we use $\tilde{\Pi}_o^{-1}(i)$ defined as $\tilde{\Pi}_o^{-1}(i) = \Pi_o^{-1}(i)$ if $\Pi_o^{-1}(i) \leq l$ and $\tilde{\Pi}_o^{-1}(i) = l+1$ otherwise.

The PSR method basically uses permutations to rank objects accordingly with the distance between permutations, rather than the original distance between objects. Therefore, in this case, pivot selection techniques should mainly aim at having the rank obtained with the permutation distance as close as possible to the rank obtained with the original distance.

4.2. MI-File

The Metric Inverted File approach (MI-File) [3, 2] uses an inverted file to store relationships between permutations. It also uses some approximations and optimizations to improve both efficiency and effectiveness.

The basic idea is that entries (the lexicon) of the inverted file are the pivots P. The posting list associated with a pivot is a list of pairs, each containing an object of the dataset and the position of the pivot in the permutation representing the object. More formally, the posting list associated with $p_i \in P$ is a list of pairs $(o, \Pi_o^{-1}(i))$, $o \in C$, i.e. a list where each object o of the dataset C is associated with the position of the pivot p_i in Π_o. For instance, an entry $(o, 7)$ in the posting list associated with the pivot p_i, indicates that p_i is the 7th closest pivot to o among those in P.

As already mentioned, in [2] it was observed that truncated permutations can be used without huge lost of effectiveness. MI-File allows truncating the permutation of both data and query objects independently. We denote with l_x the length of the permutation used for indexing and with l_s the length used for searching (i.e. the length of the query permutation). This, at the same time, offers better effectiveness, since the closest pivots are the most representative, and highest efficiency, since data object representation is

smaller and consequently objects are contained in a few posting lists making the inverted file more sparse. Using truncated permutations in the query further improving performance since just l_x posting lists are accessed rather than all posting lists.

The MI-File also uses a technique to read just a small portion of the accessed posting lists, containing the most promising objects, further reducing the search cost. The most promising data objects in a posting list, associated with a pivot p_i for a query q, are those whose position of the pivot p_i, in their associated permutation, is closer to the position of p_i in the permutation associated with q. That is, the promising objects are the objects o, in the posting list, having a small $|\Pi_o^{-1}(i) - \Pi_q^{-1}(i)|$. To control this, a parameter is used to specify a threshold on the maximum allowed position difference (mpd) among pivots in data and query objects. Provided that entries in posting lists are maintained sorted according to the position of the associated pivot, small values of mpd imply accessing just a small portion of the posting lists.

Finally, in order to improve effectiveness of the approximate search, when the MI-File execute a k-NN query, it first retrieves $k \cdot amp$ objects using the inverted file, then selects, from these, the best k objects according to the original distance. The factor $amp \geq 1$, is used to specify the size of the set of candidate objects to be retrieved using the permutation based technique, which will be reordered according to the original distance, to retrieve the best k objects.

The MI-File search algorithm computes incrementally a relaxed version of the Spearman Footrule Distance with location parameter l between the query and data objects retrieved from the read portions of the accessed posting lists.

In MI-File pivots are primarily used to identify the portions of the posting lists to access. Distances between permutations are also used to rank retrieved objects and to identify the candidate set of objects, for producing the query result. With MI-File a good selection technique should be able, at the same time, to provide a permutation distance that rank objects consistently with the original distance, and to discard pivots behaving as hot spots or pivots rarely occurring. Note that pivots behaving as hot spots, that is pivots occurring in several truncated permutations, produce very long posting lists. On the other hand, rarely occurring pivots produce very short posting lists. Posting should be balanced as much as possible.

4.3. PP-Index

The Permutation Prefix Index (PP-Index) [15, 16] associates each indexed object o with the *short* prefix Π_o^l, of length l, of the permutation Π_o, i.e., the ordered sequence of the l closest pivots.

The PP-Index belongs to the category of data structures that uses the permutations as a device to efficiently identify small sets of candidates on which to compute the distance function d. The PP-Index data structures are never used to directly approximate the distance function d, such as in the other methods presented in this section.

The permutation prefixes of the indexed objects are indexed by a *prefix tree* kept in main memory. All the relevant information relative to the indexed objects are serialized sequentially in a *data storage*, kept on disk, following the lexicographic order defined by the identifiers of the pivots in the permutation prefixes.

At search time the permutation prefix Π_q^l of the query q is used to find, in the prefix tree, the smallest subtree which includes at least $z \geq k$ candidates (z is a parameter of the search function). All the $z' \geq z$ candidates that are included in that subtree, i.e., $o_1 \dots o_{z'}$, are then retrieved from the data storage and sorted, using a max-heap of k elements, by their distance $d(q, o_i)$, thus determining the approximated k-NN result.

A key property of PP-Index is that any subtree of the prefix tree maps directly into a single sequence of *contiguous* objects in the data storage. The sequential access to secondary memory is crucial for the search efficiency. For example, in our experimental setup, a random access read from disk of the data representing 10,000 objects from the test dataset (described in Section 5.1) takes 87.4 seconds, while a sequential read of the same number of objects takes 0.14 seconds. Even using *solid storage drives* (SSD) the sequential read speed is roughly five time faster than the random access read, which takes 0.72 seconds to read 10,000 objects, due to the overhead of issuing distinct read calls. Computing 10,000 distances between objects in the test dataset takes only 0.0046 seconds, which indicates how having good secondary memory access patterns is the key aspect for efficiency.

PP-Index shares many intuitions with the Locality-Sensitive Hashing (LSH) model [20, 28]. For example, the above formulation of the search func-

An open source implementation of PP-Index is available at `http://www.esuli.it/fossil/repo/mipai`

tion that search a single subtree from the permutation prefix of the query is very fast but has a poor recall. Following the same principle of Multi-Probe LSH [24], the PP-Index adopts a *multiple-query* strategy that generates additional queries by performing local permutations on the original permutation prefix of the query object. The intuition is that the additional retrieved candidates are still close to the query because their permutation prefix differ only for a swap in a pair of adjacent pivots. Practically, the first pair of pivot identifiers that is swapped in the prefix is the one that has the minimum difference of distances with respect to the query, i.e. $\min_j(d(q, p_{\Pi_q(j+1)}) - d(q, p_{\Pi_q(j)}))$, and so on.

In the case of the PP-Index, a pivot selection technique should intuitively select a set of pivots in such a way that the size of each set of objects having the same permutation prefix has a low variance, in order to avoid cases in which the set of selected candidates is very large due to a high concentration of objects sharing the same prefix. This intuitive property is desirable to be valid not only for the full prefix length l, but also for shorter prefixes, in order to avoid issue related to a high concentration of objects for any height of the subtrees that could be selected by the search function.

4.3.1. PP-Index and M-Index

The M-Index [27] is a permutation-based similarity access method that adopts an approach of similar to the one of PP-Index. M-Index uses the permutation prefixes to compute a mapping of any object to a real number that is then used as the key to sequentially sort the indexed objects in a secondary memory data structure such as a sequential file of a B^+-tree. As for the PP-Index, the M-Index relies on the locality properties of the permutation prefixes to map similar objects to similar keys, thus enabling the efficient retrieval of candidate objects just by accessing the data structure with the key relative to the query.

5. Experiments

5.1. Datasets and Groundtruth

Experiments were conducted using the CoPhIR dataset [6], which is currently the largest multimedia metadata collection available for research purposes. It consists of a crawl of 106 millions images from the Flickr photo sharing website. We have run experiments by using as the distance function

d a linear combination of the five distance functions for the five MPEG-7 descriptors that have been extracted from each image. As weights for the linear combination we have adopted those proposed in [5] and [4]. As the ground truth, we have randomly selected 1,000 objects from the dataset as test queries and we have sequentially scanned the entire CoPhIR to compute the exact results.

5.2. Evaluation Measures

All the tested similarity search techniques re-rank a set of approximate result using the original distance. Thus, if the k-NN results list $\tilde{\mathcal{R}}_k$ returned by a search technique has an intersection with the ground truth $\mathcal{R}_k$, the objects in the intersection are ranked consistently in both lists. The most appropriate measure to use is then the *recall*: $|\tilde{\mathcal{R}}_k \cap \mathcal{R}_k|/k$. In the experiments we fixed the number of results k requested to each similarity search techniques to 100 and evaluated the *recall@r* defined as $|\tilde{\mathcal{R}}_r \cap \mathcal{R}_r|/r$ where $\tilde{R}_r$ indicates the sub-list of the first r results in $\tilde{R}_k$ ($1 \leq r \leq k$). Note that, being the two lists consistently ordered, $\tilde{\mathcal{R}}_k \cap \mathcal{R}_r \subset \tilde{\mathcal{R}}_r$ always holds and thus $\tilde{\mathcal{R}}_r \cap \mathcal{R}_r = \tilde{\mathcal{R}}_k \cap \mathcal{R}_r$, i.e. none of the results in $\tilde{\mathcal{R}}_k$ after the r-th position can give a contribute to *recall@r*. Given that the queries were selected from the dataset and that all the tested access methods always found them, we decided to remove each query from the relative approximate result list. In fact, not removing them would result in artificially raising the *recall@r* for small values of r.

The average query cost of each tested technique was measured adopting a specific cost model that will be specified in Section 5.4.

5.3. Selection Techniques Parameters

Given previous results reported in [3, 2, 15, 16] we decided to use 1,000 pivots. The parameters used for each selection technique were selected so that they required almost the same time to be computed (about 10 hours):

- FFT: We selected the pivots among a subset of 1 million randomly selected objects performing at each iteration 100,000 tries for selecting the added pivot.
- kMED: We performed the clustering algorithm on a subset of 1 million randomly selected objects.

- PSIS: We randomly selected 10,000 pairs of objects from the dataset and performed 10 trials at each iteration.
- BPP: We randomly selected a set of 10,000 candidate pivots and tested them on 100,000 randomly selected objects performing at each iteration no more than 100 trials for selecting the pivot to be removed.

5.4. Results

For all the tested similarity access methods we show a pair of figures. On the first one we report *recall@r* obtained by the various selection techniques keeping fixed the parameters of the access method settings. Even if the parameters are fixed, the use of different sets of pivots results in different average query cost which can not be inferred from this figure. For this reason, in the second figure we report an orthogonal evaluation that compares the *recall*@10 versus the query cost while varying some parameters of the access methods.

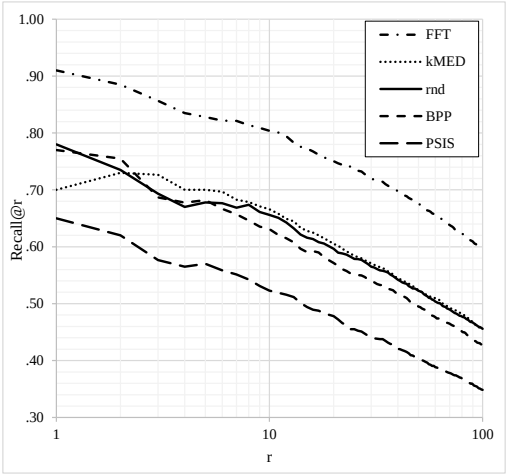

Figure 1: *Recall@r* obtained by PSR for l=100 varying r

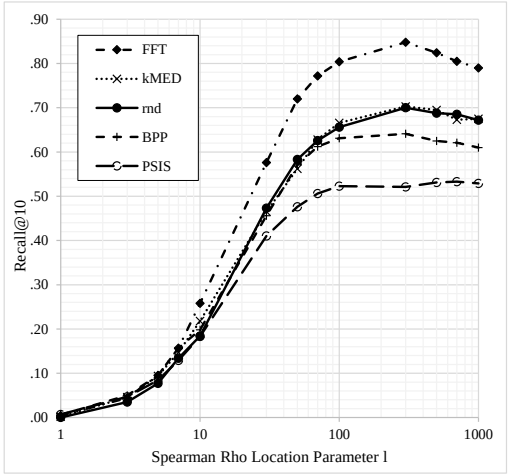

Figure 2: *Recall@10* obtained by PSR for various location parameters

5.4.1. PSR

In Figure 1 we report the *recall@r* obtained by PSR for location parameter $l = 100$. The results show that FFT outperforms the other techniques in terms of effectiveness. PSIS performs significantly worse than all the others while the rest of the techniques obtained very similar results. In Figure 2 we

tested various values of location parameter l. The choice of the parameter l directly impacts the query cost by reducing the index size and the permutation comparison cost. In fact, the PSR method performs a sequential scan of the entire set of permutations (106 millions, in these experiments), so smaller l implies less data to be accessed. The results confirm that FFT significantly outperforms the others but also reveal that the differences are more relevant when l is closer to n, i.e. when more complete permutations are used. For values of l greater than 100, none of the techniques reported significant variations. The values of l used for the results reported in Figure 1 was chosen according to this observation.

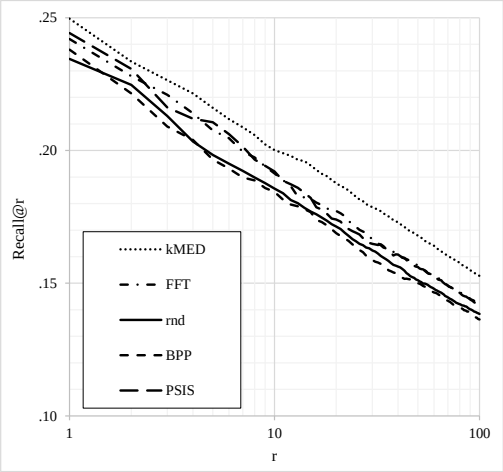

Figure 3: *Recall@r* varying r obtained by the PP-Index using the multiple-query search (eight additional queries).

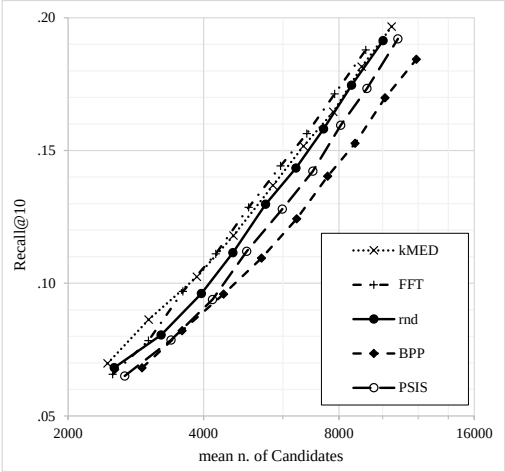

Figure 4: *Recall@10* versus the number of candidates accessed (z') by the PP-Index when using the multiple-query search method with zero (lower left corner) to eight (upper right) of additional queries.

5.4.2. PP-Index

Following the results of [16], we tested the PP-Index by setting the length of the prefixes l to 6, and the values of z to 1,000. We tested both single- and multiple-query search, exploring a range of additional queries from 1 to 8. As the reference configuration we have chosen the one using a multiple-query search method with eight additional queries (nine total).

Figure 3 shows that the PP-Index obtains its best results when using the kMED technique, which is clearly better than the other techniques. FFT

and PSIS form a group of second best techniques, followed by rnd and BPP, which are the worst performing ones. With respect to the other tested access methods, the PP-Index resulted to be more robust (or less sensitive) to the change of the pivot selection technique. The recall curves for the various techniques have an almost identical slope and there is only an average difference of 1.3% between the best and worst techniques, almost constant across all the recall levels.

For the PP-Index, we have measured the query cost induced by the various techniques in terms of number of candidate objects selected by the queries on the prefix tree. Figure 4 shows that the best two techniques with respect to the recall/cost tradeoff are kMED and FFT, followed by rnd and PSIS, with BPP being the worst one. On the nine queries setup BPP needs about 20% more candidates to score a slightly worse recall than FFT. Again, the differences between the various techniques are smaller than those observed for the other access methods, indicating a good robustness of PP-Index with respect to the pivot selection technique.

Note that the X axis of Figure 4 has a logarithmic scale. The almost straight lines indicate that the number of candidates grows with a logarithmic trend as more queries are used with the multiple-query search strategy, while the recall grows linearly, indicating that the multiple-query strategy has a very convenient recall/cost trend.

In summary, the kMED technique resulted to be the best one, resulting in higher recall at a competitive cost.

It is necessary to observe that the maximum recall obtained by PP-Index in the experiments here is 0.25, while in [16] PP-Index achieves a recall higher than 0.9. That results were obtained by using multiple indexes of the same data set, each one generated using different sets of randomly sampled pivots (multiple-index search). Results using a single index obtained in this section are indeed comparable with those obtained in [16] for the same configuration. The multiple index approach is not used here in order to maintain some uniformity among all tested access methods, and to not introduce an additional dimension that would have reduced the readability of results. Section 5.5.1 reports the results of experiments using the multiple-index search, which obtain much higher recall and confirm the observations made on the experiments of this section.

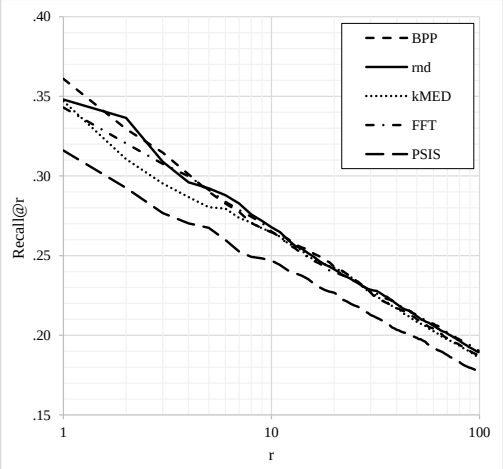

Figure 5: *Recall@r* obtained by the MI-File using $l_s = 5$, varying the number of retrieved objects r from 1 to 100.

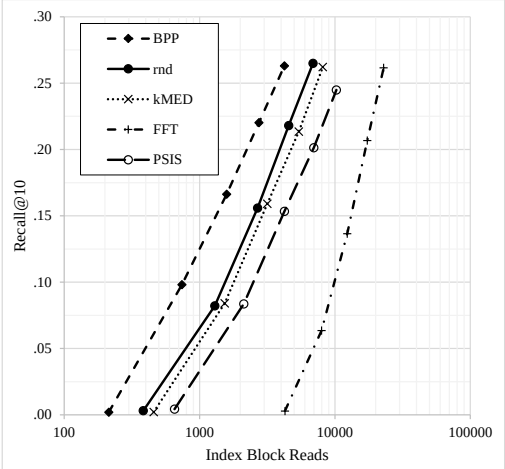

Figure 6: *Recall@10* obtained by MI-File ranging l_s from 1 to 5

5.4.3. MI-File

MI-File was tested indexing data objects using the closest 100 pivots ($l_x = 100$). Queries were executed ranging the number of closest pivots from 1 to 5, i.e. $l_s \in \{1, \ldots 5\}$ (see Section 4.2). The maximum allowed position difference among pivots in data and query objects was 5 ($mpd = 5$). The size of the set of candidate objects retrieved was set to be 50 times k, ($amp = 50$).

Figure 5 shows the results obtained using l_s fixed to 5. For $r < 10$, BPP and rnd reveal better performance, while for $r > 10$ all the techniques almost overlap, except PSIS that is always the worst.

Figure 6 shows the results varying l_s from 1 to 5. Larger values of l_s imply larger number of disk blocks reads. It can be seen that once a target recall value is fixed, the cost needed by the MI-File to achieve such recall, varies significantly among the techniques. The cost needed to achieve a specific recall using the BPP method is one order of magnitude smaller than using the FFT method. For instance, the cost needed to obtain a *recall*@10 of 0.26 is 3,000 disk block reads using BPP, while the same recall requires 25,000 disk block reads using FFT.

The BPP method is overall the one offering the best performance with MI-File. The recall value obtained using $l_s = 5$ is mostly at the top. The cost needed to execute queries is significantly lower than all the other methods.

This can be explained by the fact that, as discussed in Section 3.4, the BPP technique has been designed to distribute the positions of the various pivots uniformly across the various permutations. This means that the posting lists of the MI-File are well balanced and that they tend to contain blocks of entries, related to the same pivot position, of equal size. As a consequence, there are no posting lists that are very long and that are also mostly accessed for any query, simultaneously improving effectiveness and efficiency.

As mentioned also for the PP-Index the maximum recall obtained in these experiments is lower than that obtained in the paper where MI-File was proposed [2]. In fact here we obtain maximum recall 0.36, while in [2] recall arrives to 0.9. This is due to the fact that the total number of pivots used in these experiments is 1000. However, according to the guideline given in [2], the number of pivots should have been 20,000. This choice here was taken to maintain the same testing environment among the different methods, given that the focus was not on evaluating the access methods, but on evaluating the selection techniques. Again, Section 5.5.2 reports the results of experiments using a high-accuracy setup, which obtains a high recall still confirming the observations made on the experiments of this section.

5.4.4. Recall/cost-based evaluation

In Table 1 we report the *Recall@r/cost_unit* ratio obtained by the various access methods in conjunction with the five pivot selection techniques. The table basically aggregates and normalizes by the unit costs the results depicted in Figures 2, 4 and 6. The evaluation of the *Recall@r/cost_unit* ratio allows to measure the impact of the pivot selection techniques in terms of the efficiency they bring into the search process.

For each access methods we reported the results obtained varying a parameter (specific for each access method) that allows varying the trade-off between efficiency and effectiveness . In the table we use the random technique as the reference baseline. For the other selection techniques we report, in addition to the *Recall@r/cost_unit* ratio, also the relative percentual increment/decrement with respect to the baseline.

Results reveal that random selection of pivots is never the best choice for any of the access methods and for any of the parameters. However, there is no selection techniques that performs best for all the access methods and for the very same method we have very different results for different parameters.

For MI-File, BPP is always the best choice resulting in performance increments between 61% and 109% depending on the parameter l_s, i.e. the length

	parameter	rnd	kMED		FFT		PSIS		BPP	
PSR	*l= 3*	.0117	-	.0153	+31%	.0147	+26%	.0157	+34%	.0167	**+43%**
	l= 5	.0154	-	.0188	+22%	.0190	+23%	.0168	+9%	.0190	**+23%**
	l= 10	.0184	-	.0218	+18%	.0258	**+40%**	.0183	-1%	.0198	+8%
	l= 100	.0066	-	.0067	+2%	.0080	**+23%**	.0052	-20%	.0063	-4%
	l= 1000	.0007	-	.0007	+0%	.0008	**+18%**	.0005	-21%	.0006	-9%
PP-Index	q'= 1	.0213	-	.0244	**+14%**	.0217	+2%	.0215	+1%	.0195	-8%
	q'= 3	.0201	-	.0228	**+13%**	.0225	+12%	.0197	-2%	.0179	-11%
	q'= 5	.0196	-	.0207	+6%	.0213	**+9%**	.0182	-7%	.0161	-18%
	q'= 8	.0174	-	.0180	+3%	.0187	**+7%**	.0161	-8%	.0140	-19%
MI-File	l_s= 2	.0634	-	.0549	-13%	.0080	-87%	.0395	-38%	.1324	**+109%**
	l_s= 3	.0581	-	.0502	-14%	.0111	-81%	.0362	-38%	.1047	**+80%**
	l_s= 4	.0479	-	.0394	-18%	.0119	-75%	.0289	-40%	.0803	**+68%**
	l_s= 5	.0385	-	.0322	-16%	.0114	-70%	.0239	-38%	.0621	**+61%**

Table 1: *Recall@r/cost* obtained by the various access methods for different parameters. Cost is measured in thousands of block reads for both PP-Index and MI-File; we use permutation length l for PSR.

of the query permutation. Results also show that the longer the query permutation the less the improvement achievable using BPP with respect to random selection. These considerations are coherent with the results obtained by PSR. In fact, for small values of the permutation length l, PSR performs better when pivots are selected using BPP. Instead for larger l, FFT significantly outperforms both BPP and rnd. This confirms that, when long permutations are used, the most important aspects for PSR is the capability of the selection method to identify pivots that make the distance between permutation compatible with the original distance between objects.

PP-Index is the access method less affected by the pivot selection technique used. For small q' values (single query, three queries), a performance increment around 10% can be obtained using kMED, while for larger number of queries FFT is preferable. This indicates that kMED technique works better at assigning to similar objects the same prefix or a very similar prefix (i.e. a prefix which differ for a few swaps of pivot identifiers), thus not dispersing similar object among very different prefixes. BPP performs always worst for PP-Index, indicating that the MI-File and the PP-Index are two inherently

different data structures, even though they intuitively seem to require similar properties from the pivot sets, i.e. uniform distribution of object among posting lists/permutation prefixes.

5.5. High accuracy experiments

The goal of the experiments in previous section is to compare the pivot selection techniques on each access method. In this scenario achieving top accuracy is not of key importance (we do not need to compare the access methods between themselves), as long as the factors that determine the higher/lower accuracy of an access method are not influenced by the pivot selection technique in use. Our experimental setup for experiments reported in previous sections has been designed to minimize their dependence of access method-specific parameters, in order to have a kind of "reference" setup, which could be eventually extended to additional access-methods. In this section we present results for high-accuracy configurations for the PP-Index and the MI-File, with the goal of showing that the differences observed on the reference setup among the pivot selection techniques are still present in fine-tuned setups.

5.5.1. PP-Index

For the PP-Index, the high-accuracy configuration uses four indexes and the multiple-index search strategy. All the techniques presented in this paper can be easily adapted to support the multiple-index search, i.e., by using different initial random samples of candidate pivots on which to apply the pivot selection techniques. We compare only rnd and kMED in this high-accuracy setup, i.e., the most commonly used selection strategy and the one that performed best in the experiments of Section 5.4.2. The results in the high accuracy configuration (Figure 7) confirm the previous results (Figure 3), with PP-Index obtaining a significant average Recall improvement of 1.4%, which is perfectly in line with the 1.3% value measured in experiments of Section 5.4.2. The difference between kMED and rnd is statistically significant (two-tails paired t-test) with $p < 0.001$. The cost of search plots (Figure 8) follows a similar trend to the one of the previous results (Figure 4), with the two curves overlapping for the search configurations that use a higher number of multiple queries, with a slightly smaller cost for rnd with respect to kMED but also scoring a lower recall.

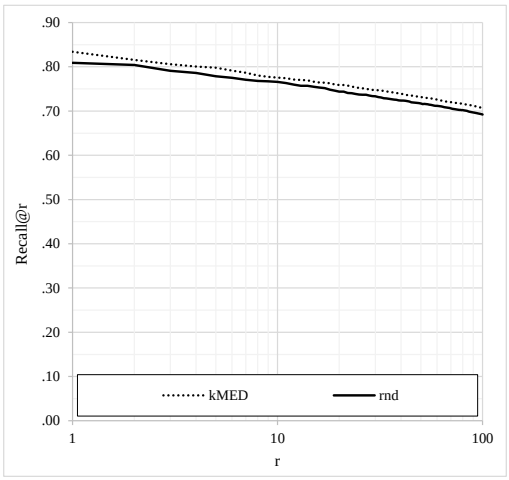

Figure 7: *Recall@r* varying r obtained by the high-accuracy PP-Index setup using the multiple-query (eight additional queries) and the multiple-index (four indexes) search.

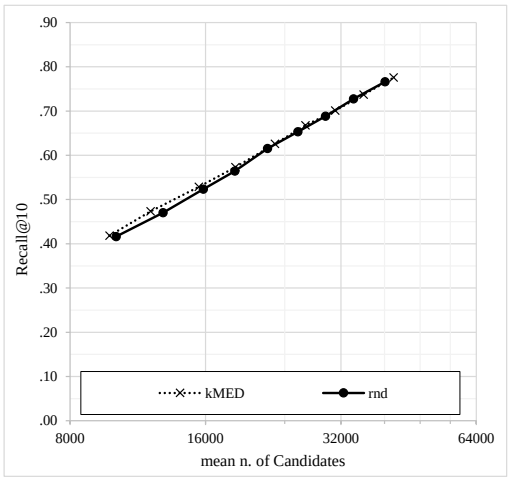

Figure 8: *Recall@10* versus the number of candidates accessed (z') by the PP-Index when using the multiple-index (four indexes) search and varying the multiple-query search method from zero (lower left corner) to eight (upper right) additional queries.

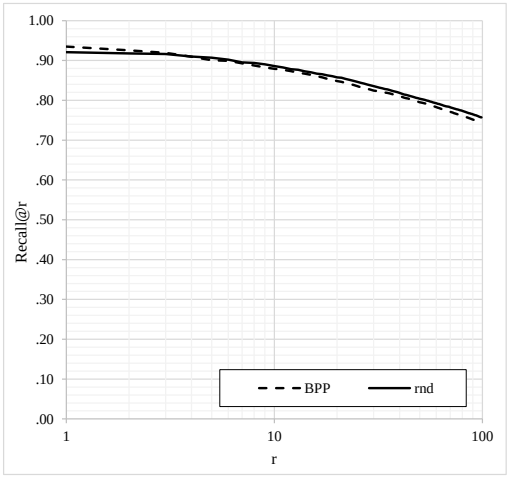

Figure 9: *Recall@r* obtained by the high-accuracy MI-File setup (20,000 pivots) using $l_s = 5$, varying the number of retrieved objects r from 1 to 100.

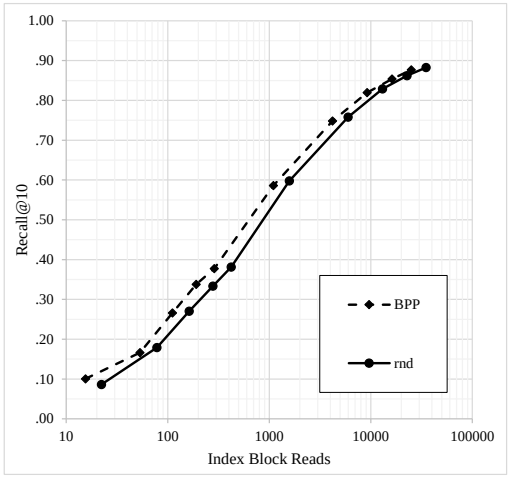

Figure 10: *Recall@10* obtained by high-accuracy MI-File setup (20,000 pivots) ranging l_s from 1 to 5

5.5.2. MI-File

In order to setup the MI-File, for high accuracy and high efficiency performance, we followed the guidelines given in [2]. The total number of pivots was set to 20,000. The dataset was indexed using the closest 100 pivots ($l_s = 100$). Queries were executed varying the number of closest pivots from 1 to 50 ($l_s \in \{1 \dots 50\}$). Parameters *mpd* and *amp* were set respectively as $mpd = l_s$ and $amp = 50$. We tested the rnd and BPP methods, to have a comparison between the baseline method and the one offering the best performance with MI-File in the experiments of Section 5.4.3.

Results are shown in Figures 9 and 10. Figure 9 shows the *recall@r* obtained by both rnd and BPP methods using l_s set to 50. In both cases recall is higher than 0.9 for small values of r, and is about 0.75 when r is 100. The curves are almost overlapped, as already seen in tests using only 1,000 pivots, with BPP being a bit higher for small values of r. However, we need to stress again that this plot just shows the recall obtained without taking into account the cost required to obtain it. Cost, in terms of number of disk block reads, is taken into consideration in Figure 10. In that figure l_s ranges from 1 up to 50, showing the different costs with respect to the relative recalls. From that graph it can be seen that for obtaining a certain recall the rnd method requires about 20% more disk block reads than the BPP method. This again confirms observation made in previous experiments with 1,000 pivots.

5.5.3. Variance of Recall

For the high-accuracy experiments we have also measured the variance of the Recall values across the 1,000 tested queries. The variance values are plotted in Figures 11 and 12, and show that rnd produces higher variance results with respect to kMED for PP-Index and BPP for MI-File. This supports the intuition that random selection of pivots may not cover the data in an optimal way, thus resulting in more variability in the accuracy obtained when executing different queries. The difference in variance is particularly noticeable for small values of r, while it tends to disappear for large values of r. The users' perception of the relevance of errors is typically higher for the first results of a query, rather than for elements with lower ranks. In fact, users typically tend to consider negatively the fact that the elements they are searching for do not appear in the first positions of the result list. Therefore, having lower variance in accuracy, specially for small values of r is a nice property, and kMED and BPP are better at this.

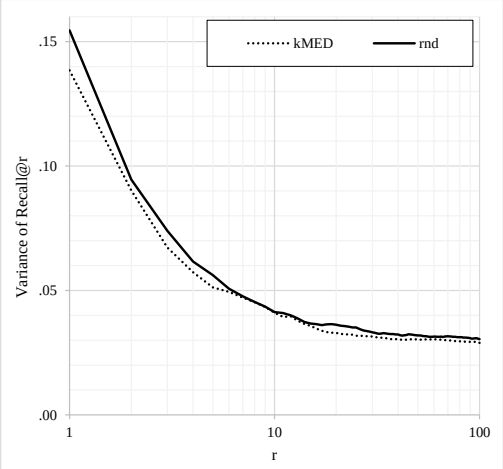

Figure 11: Variance of the *Recall@r* value across the tested 1,000 queries, using the high-accuracy configuration of PP-Index.

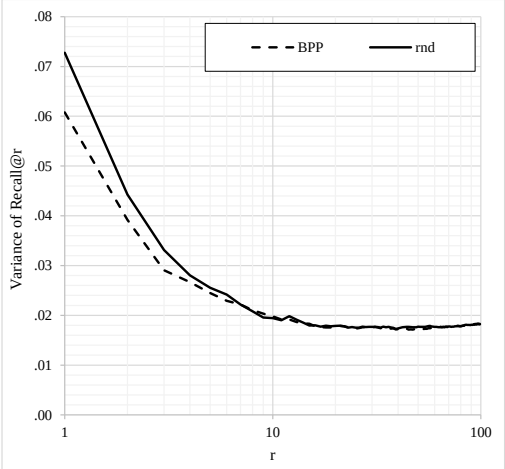

Figure 12: Variance of the *Recall@r* value across the tested 1,000 queries, using the high-accuracy configuration of MI-File.

6. Conclusion

In this paper we compared five pivot selection techniques on three permutation-based access methods. For all the tested access methods we found at least one technique that significantly outperforms the random selection. Another interesting point is that there is not a technique that is universally the best for all the access methods.

In Section 3, we identified two roles that the permutations can play in the access methods. First, they can be used for approximating the original distance between two objects by comparing their permutations (e.g., PSR). Second, the can be used to focus accessed access in the database (e.g., PP-Index). A combination of the two roles is also possible (e.g., MI-File). The tested pivots selections strategies may have the objective of obtaining a good distance approximation (e.g., FTT), helping to identify a good set of candidate objects (e.g., kMED) or both (e.g., our BPP novel technique). The results are consistent with this classification, i.e., the selection strategies that have objectives consistent with a specific access method outperforms the strategies that have different goals. It also resulted that the random chosen pivots is never a bad idea even if it is also never the smartest decision.

The PSR method, i.e. the sequential scan of the permutations adopting the Spearman Rho with location parameter l distance, largely benefited from

the use of FFT. We believe this is because PSR only rely on permutations to approximate the original distance.

The BPP technique significantly outperforms the others when used in combination with the MI-File. The benefit of the balancing is more relevant when posting lists, created relying on the permutations, are used to select good candidate sets in conjunction with similarity between permutations is used to approximate original distance between objects.

For PP-Index the best technique is kMED. We believe that kMED helps in selecting good pivots in order to find good candidate sets while not relying on the similarity between permutations. It is worth to note that BPP, the best technique for MI-File, is always the worst technique for PP-Index.

In conclusion, even if all the tested access methods belong to the class of the permutation-based methods, each of them has inherent and significantly different characteristics in the way they exploit the permutation space. Similarly, the various pivot selection techniques presented in this paper have different goals when selecting their pivots. The results of the experiments indicates that the pivot selection technique should be considered as an integrating and relevant part of an access method.

Acknoledgments

This work was partially supported by EAGLE (Europeana network of Ancient Greek and Latin Epigraphy, co-founded by the European Commision, CIP-ICT-PSP.2012.2.1 - Europeana and creativity, Project Reference 325122) and Secure! (Piattaforma intelligente basata su tecnologie crowdsourcing e crowdsensing per la sicurezza e la gestione delle crisi ed emergenze, Regione Toscana POR CReO 2007 2013, Linea di Intervento 1.5.a, 1.6, Bando Unico R&S Anno 2012).

References

[1] Giuseppe Amato, Andrea Esuli, and Fabrizio Falchi. Pivot selection strategies for permutation-based similarity search. In Nieves Brisaboa, Oscar Pedreira, and Pavel Zezula, editors, *Similarity Search and Applications*, volume 8199 of *Lecture Notes in Computer Science*, pages 91–102. Springer Berlin Heidelberg, 2013.

[2] Giuseppe Amato, Claudio Gennaro, and Pasquale Savino. Mi-file: Using inverted files for scalable approximate similarity search. *Multimedia Tools and Applications- An International Journal*, (Online first), November 2012 2012.

[3] Giuseppe Amato and Pasquale Savino. Approximate similarity search in metric spaces using inverted files. In *Proceedings of the 3rd international conference on Scalable information systems*, InfoScale '08, pages 28:1–28:10, ICST, Brussels, Belgium, Belgium, 2008. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

[4] Michal Batko, Fabrizio Falchi, Claudio Lucchese, David Novak, Raffaele Perego, Fausto Rabitti, Jan Sedmidubsky, and Pavel Zezula. Building a web-scale image similarity search system. *Multimedia Tools and Applications*.

[5] Michal Batko, Petra Kohoutkova, and Pavel Zezula. Combining metric features in large collections. In *Proceedings of the First International Workshop on Similarity Search and Applications*, SISAP '08, pages 79–86, Cancún, Mexico, 2008.

[6] Paolo Bolettieri, Andrea Esuli, Fabrizio Falchi, Claudio Lucchese, Raffaele Perego, Tommaso Piccioli, and Fausto Rabitti. Cophir: a test collection for content-based image retrieval. *CoRR*, abs/0905.4627, 2009.

[7] Tolga Bozkaya and Meral Ozsoyoglu. Indexing large metric spaces for similarity search queries. *ACM Trans. Database Syst.*, 24(3):361–404, September 1999.

[8] Sergey Brin. Near neighbor search in large metric spaces. In *VLDB'95, Proceedings of 21th International Conference on Very Large Data Bases, September 11-15, 1995, Zurich, Switzerland*, pages 574–584. Morgan Kaufmann, 1995.

[9] B. Bustos, G. Navarro, and E. Chavez. Pivot selection techniques for proximity searching in metric spaces. In *Computer Science Society, 2001. SCCC '01. Proceedings. XXI Internatinal Conference of the Chilean*, pages 33–40.

[10] B. Bustos, O. Pedreira, and N. Brisaboa. A dynamic pivot selection technique for similarity search. In *Data Engineering Workshop, 2008. ICDEW 2008. IEEE 24th International Conference on*, pages 394–401, 2008.

[11] Benjamin Bustos, Gonzalo Navarro, and Edgar Chávez. Pivot selection techniques for proximity searching in metric spaces. *Pattern Recogn. Lett.*, 24(14):2357–2366, October 2003.

[12] Edgar Chávez, Karina Figueroa, and Gonzalo Navarro. Effective proximity retrieval by ordering permutations. *IEEE Trans. Pattern Anal. Mach. Intell.*, 30(9):1647–1658, 2008.

[13] Sanjoy Dasgupta. Performance guarantees for hierarchical clustering. In *15th Annual Conference on Computational Learning Theory*, pages 351–363. Springer, 2002.

[14] Agni Delvinioti, Hervé Jégou, Laurent Amsaleg, and Michael Houle. Image retrieval with reciprocal and shared nearest neighbors. In *VISAPP 2014 - Proceedings of the 9th International Conference on Computer Vision Theory and Applications, Volume 2, Lisbon, Portugal, 5-8 January, 2014*, pages 321–328, 2014.

[15] Andrea Esuli. Mipai: Using the pp-index to build an efficient and scalable similarity search system. In *SISAP*, pages 146–148, 2009.

[16] Andrea Esuli. Use of permutation prefixes for efficient and scalable approximate similarity search. *Information Processing & Management*, 48(5):889 – 902, 2012.

[17] Ronald Fagin, Ravi Kumar, and D. Sivakumar. Comparing top k lists. In *Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms*, SODA '03, pages 28–36, Philadelphia, PA, USA, 2003. Society for Industrial and Applied Mathematics.

[18] András Faragó, Tamás Linder, and Gábor Lugosi. Fast nearest-neighbor search in dissimilarity spaces. *IEEE Trans. Pattern Anal. Mach. Intell.*, 15(9):957–962, 1993.

[19] Arthur Flexer and Dominik Schnitzer. Can shared nearest neighbors reduce hubness in high-dimensional spaces? In *13th IEEE International*

Conference on Data Mining Workshops, ICDM Workshops, TX, USA, December 7-10, 2013, pages 460–467, 2013.

[20] Aristides Gionis, Piotr Indyk, and Rajeev Motwani. Similarity search in high dimensions via hashing. In *Proceedings of 25th International Conference on Very Large Data Bases*, VLDB '99, pages 518–529, 1999.

[21] Teofilo F. Gonzalez. Clustering to minimize the maximum intercluster distance. *Theor. Comput. Sci.*, 38:293–306, 1985.

[22] Michael E. Houle, Hans-Peter Kriegel, Peer Kröger, Erich Schubert, and Arthur Zimek. Can shared-neighbor distances defeat the curse of dimensionality? In *Scientific and Statistical Database Management, 22nd International Conference, SSDBM 2010, Heidelberg, Germany, June 30 - July 2, 2010. Proceedings*, pages 482–500, 2010.

[23] L. Kaufman and P. J. Rousseeuw. *Finding groups in data: an introduction to cluster analysis*. John Wiley and Sons, New York, 1990.

[24] Qin Lv, William Josephson, Zhe Wang, Moses Charikar, and Kai Li. Multi-probe lsh: efficient indexing for high-dimensional similarity search. In *Proceedings of the 33rd International Conference Very Large Data Bases*, VLDB '07, pages 950–961, Vienna, Austria, 2007.

[25] Rui Mao, Willard L. Miranker, and Daniel P. Miranker. Dimension reduction for distance-based indexing. In *Proceedings of the Third International Conference on SImilarity Search and APplications*, SISAP '10, pages 25–32, New York, NY, USA, 2010. ACM.

[26] María Luisa Micó, José Oncina, and Enrique Vidal. A new version of the nearest-neighbour approximating and eliminating search algorithm (aesa) with linear preprocessing time and memory requirements. *Pattern Recogn. Lett.*, 15(1):9–17, January 1994.

[27] David Novak, Michal Batko, and Pavel Zezula. Metric index: An efficient and scalable solution for precise and approximate similarity search. *Inf. Syst.*, 36(4):721–733, June 2011.

[28] David Novak, Martin Kyselak, and Pavel Zezula. On locality-sensitive indexing in generic metric spaces. In *Proceedings of the Third International Conference on SImilarity Search and APplications*, SISAP '10, pages 59–66, New York, NY, USA, 2010. ACM.

[29] Rodrigo Paredes and Gonzalo Navarro. Optimal incremental sorting. In *In Proc. 8th Workshop on Algorithm Engineering and Experiments (ALENEX*, pages 171–182. SIAM Press, 2006.

[30] Marco Patella and Paolo Ciaccia. Approximate similarity search: A multi-faceted problem. *J. of Discrete Algorithms*, 7(1):36–48, March 2009.

[31] Oscar Pedreira and Nieves R. Brisaboa. Spatial selection of sparse pivots for similarity search in metric spaces. In *Proceedings of the 33rd conference on Current Trends in Theory and Practice of Computer Science*, SOFSEM '07, pages 434–445, Berlin, Heidelberg, 2007. Springer-Verlag.

[32] Danfeng Qin, Stephan Gammeter, Lukas Bossard, Till Quack, and Luc J. Van Gool. Hello neighbor: Accurate object retrieval with k-reciprocal nearest neighbors. In *The 24th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2011, Colorado Springs, CO, USA, 20-25 June 2011*, pages 777–784, 2011.

[33] Milos Radovanovic, Alexandros Nanopoulos, and Mirjana Ivanovic. Hubs in space: Popular nearest neighbors in high-dimensional data. *Journal of Machine Learning Research*, 11:2487–2531, 2010.

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

[35] Dominik Schnitzer, Arthur Flexer, Markus Schedl, and Gerhard Widmer. Local and global scaling reduce hubs in space. *Journal of Machine Learning Research*, 13:2871–2902, 2012.

[36] Dominik Schnitzer, Arthur Flexer, and Nenad Tomasev. A case for hubness removal in high-dimensional multimedia retrieval. In *Advances in Information Retrieval - 36th European Conference on IR Research, ECIR 2014, Amsterdam, The Netherlands, April 13-16, 2014. Proceedings*, pages 687–692, 2014.

[37] Marvin Shapiro. The choice of reference points in best-match file searching. *Commun. ACM*, 20(5):339–343, May 1977.

[38] Peter N. Yianilos. Data structures and algorithms for nearest neighbor search in general metric spaces. In *Proceedings of the fourth annual ACM-SIAM Symposium on Discrete algorithms*, SODA '93, pages 311–321, Philadelphia, PA, USA, 1993. Society for Industrial and Applied Mathematics.

[39] Pavel Zezula, Giuseppe Amato, Vlastislav Dohnal, and Michal Batko. *Similarity Search - The Metric Space Approach*, volume 32 of *Advances in Database Systems*. Kluwer, 2006.

[40] Pavel Zezula, Pasquale Savino, Giuseppe Amato, and Fausto Rabitti. Approximate similarity retrieval with m-trees. *The VLDB Journal*, 7(4):275–293, December 1998.