Pivot Selection Strategies for Permutation-Based Similarity Search

Giuseppe Amato, Andrea Esuli, and 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. This, of course assumes 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 strategies 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 pivots selection strategies on three permutation-based similarity access methods. Among those, we propose a novel strategy specifically designed for permutations. Two significant observations emerge from our tests. First, random selection is always outperformed by at least one of the tested strategies. Second, there is not a strategy that is universally the best for all permutation-based access methods; rather different strategies are optimal for different methods.

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).

Permutation-based indexes have been proposed as a new approach to approximate similarity search [1, 8, 11, 20]. In permutation-based indexes, data objects and queries are represented as appropriate permutations of a set of 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 measured by the effectiveness and efficacy of the resulting index structure at search time.

The paper is structured as follows. In Section 2 we discuss related work. Section 3 presents the strategies 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 strategies for access methods usually classified as *pivot-based* [25] 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 for reducing the set of data objects accessed during similarity query execution. In an early work by Shapiro [23], it was noticed that good performance were obtained by locating pivots *far away* from data clusters. In [24, 18, 5], following this intuition, several heuristics were proposed to select pivots between the *outliers* and far away from each other. Pivot selection techniques that maximize the mean of the distance distribution in the pivoted space were exploited in [7]. It was also argued that while good pivots are usually outliers, the reverse is not true. In [22, 6], the problem of dynamic pivot selection as the database grows is faced. In [17] 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.

Works that use permutation-based indexing techniques have mostly performed a random selection of pivots [1, 8, 11] 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 and also because the use of previous selection strategies 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 [8], were 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 Strategies

Permutation based access methods use pivots to build permutation that represent data objects. This paper compares four promising selection strategies used in combination with different permutation based indexes to make a comprehensive evaluation, and also to identify the specific features that can be exploited in the various cases.

As the baseline we tested the **random (rnd)** strategy, which samples pivots from the dataset following a uniform probability distribution.

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 [9]) 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 $\min_{1 \leq j \leq i} d(p_i, p_j)$ is maximum. In [14], it has been proved 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. The computational cost of this algorithm is $O(n|C|)$, where n is the number of requested pivots.

3.2 *k*-medoids (kMED)

Originally proposed in [15], 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.

3.3 Pivoted Space Incremental Selection (PSIS)

In [7] several strategies for selecting pivots were proposed and tested considering the average distance of the transformed space obtained leveraging on the set of selected pivots and on the triangle inequality of the original metric space [25]. The presented algorithms try to maximize the average distance in the pivoted space defined as: $D_P(x, y) = \max_{1 \leq i \leq n} |d(x, p_i) - d(y, p_i)|$.

The goal of the proposed algorithm is to have good lower bounds D_P for the original distance d. Bustos et al. observed that the chosen pivots are outliers but

that not all outliers are good pivots for maximizing the average D_P. The overall best between the proposed methods 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 occurences (BPP)

While the other pivot selection approaches mainly originate from the literature on similarity search in metric spaces and clustering, in this section we propose an algorithm specifically intended for permutation-based access methods. The intuition suggests that each pivot should appear in the various positions in the permutations uniformly. In fact, if a pivot $p_i \in P$ appears in the same position in all the permutations, such pivot is useless.

Let $c(p_i, j) = |\{\Pi_o : \Pi_o^{-1}(i) = j\}|$ be the number of permutations where p_i appears in position j. 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 $|C|/n$. BPP tries to minimize the deviation of $c(p_i, j)$ values from their mean. 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 [1] it was observed that the first pivots in the permutations, i.e. the nearest to the object, have been proved to be 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. 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.

4 Similarity Access Methods

We have compared the pivot selection strategies on three permutation based index structures that reasonably cover the various approaches adopted in literature by 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 [8]. As distance between permutations, Spearman Rho, Kendall Tau and Spearman Footrule [12] were tested. Spearman Rho revealed better performance. Given two permutations Π_x and Π_y, Spearman Rho is defined as:

$$S_\rho(\Pi_x, \Pi_y) = \sqrt{\sum_{1 \leq i \leq n} (\Pi_x^{-1}(i) - \Pi_y^{-1}(i))^2}$$

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 S_ρ (in our experiments we fixed $k' = 10k$). This set is then reordered considering the original distance d. In [8] an optimal incremental sorting [21] 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, the most relevant information of the permutation Π_o is in the very first, i.e. nearest, pivots. Thus, we decided to test also truncated permutations. In this case we used the Spearman Rho distance with location parameter $S_{\rho,l}$ defined in [12], which is intended for the comparison of top-l lists. $S_{\rho,l}$ differs from S_ρ for the use of an inverted truncated permutation $\tilde{\Pi}_o^{-1}$ that assumes that pivots further than $p_{\Pi_o(l)}$ from o being at position $l+1$. Formally, $\tilde{\Pi}_o^{-1}(i) = \Pi_o^{-1}(i)$ if $\Pi_o^{-1}(i) \leq l$ and $\tilde{\Pi}_o^{-1}(i) = l+1$ otherwise.

4.2 MI-File

The Metric Inverted File approach (MI-File) [2, 1] 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 an entry $p_i \in P$ 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.

As already mentioned, in [1] 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 one used for searching (i.e. the length of the query permutation).

The MI-File also uses a strategy 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 Footrule Distance with location parameter l between the query and data objects retrieved from the read portions of the accessed posting lists.

4.3 PP-Index

The Permutation Prefix Index (PP-Index) [10, 11] associates each indexed object o with the *short* prefix Π_o^l, of length l, of the permutation Π_o. The permutation prefixes of the indexed objects are indexed by a *prefix tree* kept in main memory. All the indexed objects are serialized sequentially in a *data storage*, kept on disk, following the lexicographic order defined by 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 disk is crucial for the search efficiency. For example, in our experimental setup, random access read from disk of 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. Computing 10,000 distances between objects in the test dataset takes only 0.0046 seconds, which indicates how having good disk access patterns is the key aspect for efficiency.

The approach of PP-Index to similarity search is close to the one of M-Index [19] , which uses 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.

Both PP-Index and M-Index share many intuitions with the Locality-Sensitive Hashing (LSH) model [13, 20]. For example, following the same principle of Multi-Probe LSH [16], 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, i.e. retrieving additional candidates that are still close to the query because their permutation prefix differ only for a swap in a pair of adjacent pivots. The first pair that is swapped 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. Note that it may happen that some of the additional queries end up in selecting the same subtree of other queries, so that the number of sequences of candidates objects accessed on disk may be less than the number of queries.

5 Experiments

5.1 Experimental Settings

Datasets and Groundtruth: Experiments were conducted using the CoPhIR dataset [4], 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 [3]. 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.

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 5.2.

Selection Techniques Parameters: Given previous results reported in [2, 1, 10, 11] we decided to use 1,000 pivots. The parameters used for each selection strategy 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.

5.2 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 strategies keeping 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.

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 strategies obtained very similar results. In Figure 2 we tested various values of location parameter l, which directly impacts the query cost by reducing the index size and the permutation comparison cost. 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.

PP-Index: Following the results of [11], 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). As already noted in Section 4.3, some of the additional queries may result in selecting the same subtree of candidates. In fact, only 4.61 sequential blocks of candidates are accessed on disk on average for the above configuration.

Figure 3 shows that the PP-Index obtains its best results when using the kMED strategy, which is clearly better than the other strategies. FFT and PSIS form a group of second best strategies, 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 strategy. The recall curves for the various strategies have an almost identical slope and there is only an average difference of 1.3% between the best and worst strategies, almost constant across all the recall levels.

For the PP-Index, we have measured the query cost induced by the various strategies in terms of number of candidate objects selected by the queries on the prefix tree. Figure 4 shows that the best two strategies 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 strategies are smaller than those observed for the other access methods.

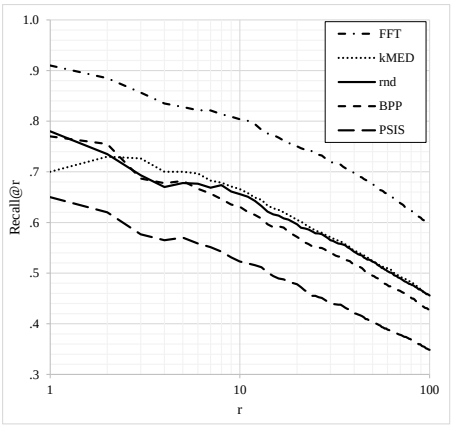

Fig. 1. *Recall@r* obtained by PSR for l=100 varying r

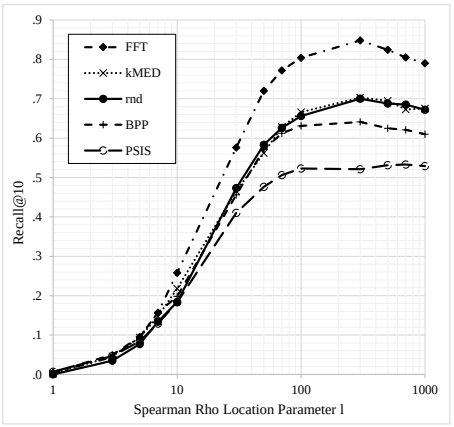

Fig. 2. *Recall@10* obtained by PSR for various location parameters

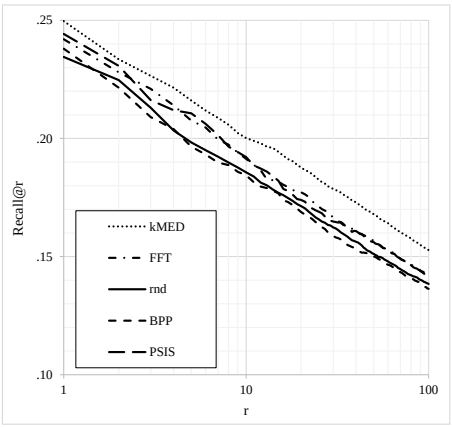

Fig. 3. *Recall@r* varying r obtained by the PP-Index using the multiple-query search (eight additional queries).

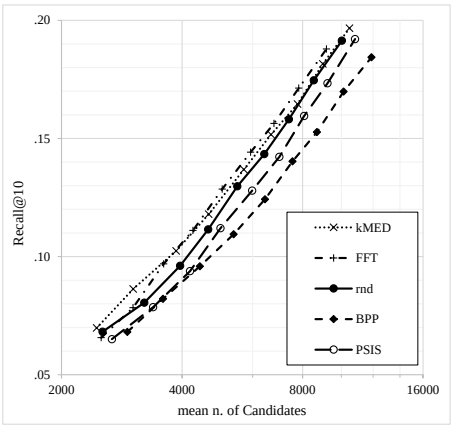

Fig. 4. *Recall@10* versus the number of candidates accessed (z') by the PP-Index when using the multiple-query search method with zero (lower left corner) to eight (upper right) of additional queries.

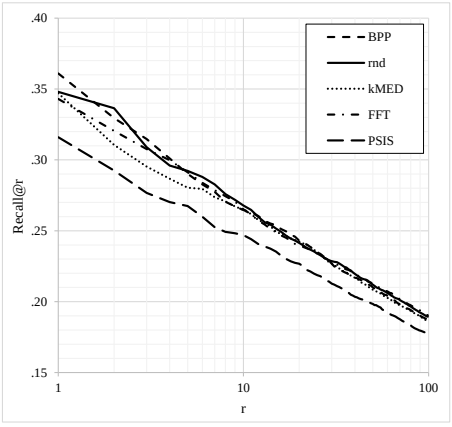

Fig. 5. *Recall@r* obtained by the MI-File using $l_s = 5$, varying the number of retrieved objects r from 1 to 100.

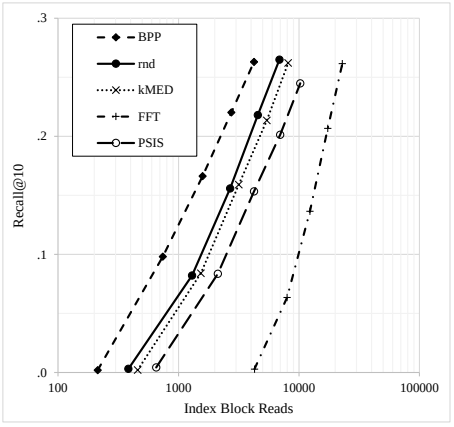

Fig. 6. *Recall@10* obtained by MI-File ranging l_s from 1 to 5

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 strategy resulted to be the best one, resulting in higher recall at a competitive cost.

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. $l_s \in \{1, \dots 5\}$ (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 strategies 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 strategies. 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 strategy 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.

6 Conclusion and Future Work

In this paper we compared five pivots selection strategies on three permutation-based access methods. For all the tested access methods we found at least one strategy that significantly outperforms the random selection. Another interesting point is that there is not a strategy that is universally the best for all the access methods. 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. For PP-Index the best strategy has been kMED even if the performance differences are small. The novel proposed BPP strategy significantly outperformed the others when used in combination with the MI-File.

This means that even if all the tested access methods are permutation-based, they significantly differ in the way they exploit the permutation space.

The CoPhIR collection is one of largest non-synthetic collections available for experiments on similarity search and its objects have a relatively high dimensionality. The results we have observed on this collection should thus be a good reference for practical applications that have similar characteristics (e.g., large collections of images). We are planning to extend the comparison on other collections with different characteristics in terms of data type, collection size and dimensionality. For the future we also plan to expand the comparison to other data structures, such as the M-Index [19], and to test novel strategies that make use of information on the queries, e.g., from a query log (as suggested in [11]).

References

1. 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.
2. 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).
3. 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*.
4. 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.
5. 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.
6. 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.
7. 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.
8. Edgar Chávez, Karina Figueroa, and Gonzalo Navarro. Effective proximity retrieval by ordering permutations. *IEEE Trans. Pattern Anal. Mach. Intell.*, 30(9):1647–1658, 2008.
9. Sanjoy Dasgupta. Performance guarantees for hierarchical clustering. In *15th Annual Conference on Computational Learning Theory*, pages 351–363. Springer, 2002.
10. Andrea Esuli. Mipai: Using the pp-index to build an efficient and scalable similarity search system. In *SISAP*, pages 146–148, 2009.
11. Andrea Esuli. Use of permutation prefixes for efficient and scalable approximate similarity search. *Information Processing & Management*, 48(5):889 – 902, 2012.

12. 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.
13. 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.
14. Teofilo F. Gonzalez. Clustering to minimize the maximum intercluster distance. *Theor. Comput. Sci.*, 38:293–306, 1985.
15. L. Kaufman and P. J. Rousseeuw. *Finding groups in data: an introduction to cluster analysis*. John Wiley and Sons, New York, 1990.
16. 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.
17. 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.
18. 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.
19. 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.
20. 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.
21. 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.
22. 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.
23. Marvin Shapiro. The choice of reference points in best-match file searching. *Commun. ACM*, 20(5):339–343, May 1977.
24. 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.
25. Pavel Zezula, Giuseppe Amato, Vlastislav Dohnal, and Michal Batko. *Similarity Search - The Metric Space Approach*, volume 32 of *Advances in Database Systems*. Kluwer, 2006.