【电子与信息科学 / Electronics and Information Science】 ·人工智能与数字经济专题·

RPA: a memory-efficient metric-space recall@*R* ANNS index

JIANG Runben, CHEN Jiaying, and MAO Rui

College of Computer Science and Software Engineering, Guangdong Provincial Key Laboratory of Popular High Performance Computers, Guangdong Provincial Engineering Center of China-made High Performance Data Computing System, Shenzhen Key Laboratory of Service Computing and Applications, Shenzhen University, Shenzhen 518060, Guangdong Province, P. R. China

Abstract: Approximate nearest neighbor search (ANNS) for high dimensional data has received extensive research efforts. Many existing ANNS methods in metric spaces are essentially based on the permutation of pivots, or pre-selected reference points, which are ordered by their distances to the data. Unfortunately, most of these permutation-based methods are quite memory demanding, because of either complexity of index structure, excessive pivots, or insufficient exploitation of distances. This paper proposes the reduced permutation array (RPA), a delicate integration of existing technical elements based on in-depth understanding, for recall@R ANNS in metric spaces. As an array, for each data element, RPA stores only l identities (IDs) of its nearest pivots out of k pre-selected ones ($l \ll k$). During search, the order of distances from each data element to the query is approximated by a score function based on the distances from its nearest pivots to the query, and a bounded heap is maintained to store R candidate results. RPA is simple, memory-efficient, and scalable. Experiments show that RPA is approximately 3 times more memory-efficient than existing methods on average. The research results can provide an effective ANNS method for massive data in a single-machine environment with limited memory resources.

Key words: computer science and technology; approximate nearest neighbor search; metric space; index structure; pivot selection; pivot permutation; memory-efficient

CLC number: TP301.6 **Document code:** A **DOI:** 10.3724/SP.J.1249.2023.06640

RPA：一种内存高效的度量空间recall@*R*近似最近邻搜索索引

江润本，陈家颖，毛睿

深圳大学计算机与软件学院，广东省普及型高性能计算机重点实验室，广东省国产高性能数据计算系统工程技术研究中心，深圳市服务计算与应用重点实验室，广东深圳 518060

摘 要：现有的度量空间的近似最近邻搜索(approximate nearest neighbor search，ANNS)方法通常依赖于预选择的支撑点构成的序列，序列中的支撑点按照到数据元素的距离升序排列．然而，大多数现有的度量空间ANNS方法由于索引结构复杂、支撑点过多或者未能充分利用距离信息导致搜索时内存开销巨大．为此，提出精简排列阵(reduced permutation array，RPA)的度量空间recall@R近似最近邻搜索方法．对于全体数据元素，RPA预先选择k个支撑点，对每个数据元素仅存储离该数据元素最近的l个($l \ll k$)，并将所有

Received: 2023-07-03; **Accepted:** 2023-08-28; **Online** (CNKI): 2023-10-12
Foundation: National Natural Science Foundation of China (62072311)
Corresponding author: Professor MAO Rui (mao@szu.edu.cn)
Citation: JIANG Runben, CHEN Jiaying, MAO Rui. RPA: a memory-efficient metric-space recall@*R* ANNS index [J]. Journal of Shenzhen University Science and Engineering, 2023, 40(6): 640-648.

http://journal.szu.edu.cn

元素的支撑点序列构建为一个数组结构．在搜索过程中，利用一种得分函数，该函数基于查询对象到各个支撑点的距离来近似计算数据元素到查询对象的距离．同时，维护一个有界最小堆，以保存R个候选结果数据元素．RPA具有结构简单、内存效率高和可扩展性强等特点．实验结果表明，在相同召回率的情况下，与排列索引(permutation-based index，P-index)相比，RPA平均具有高达3倍的内存压缩比．研究结果可在内存资源有限的单机环境下提供一种有效的针对海量数据的ANNS方法．

关键词：计算机科学与技术；近似最近邻搜索；度量空间；索引结构；支撑点选择；支撑点序列；内存高效

In the last few decades, approximate nearest neighbor search (ANNS)[1] has received more and more attention. A common definition of ANNS is to return a data element that is within $(1+\epsilon)d$ distance to the query, where d is the distance from the query to its nearest neighbor[2] and ϵ is a loose coefficient, intended to tolerate a certain loss of accuracy while obtaining higher search efficiency. Widely adopted in real applications such as search engines, another way to conveniently measure the accuracy of ANNS is recall@R, the probability that the top R results returned contain the nearest neighbor across all queries[3-4]. Since most conclusions for retrieving single nearest neighbor hold for multiple nearest neighbors[3-4], this paper focuses on recall@R for retrieving single nearest neighbor. Further, due to the variety challenge of big data, data processing in general metric spaces[5] has been a hot topic[6-9]. This paper considers only ANNS in metric spaces.

Common metric-space ANNS methods are graph-based or permutation-based. Essentially, graph-based methods[10-17] greedily traverse the nearest-neighbor graph[16] of data. They compute the distance between query and data during searching, and thus usually have to load data and a huge nearest-neighbor graph into memory, resulting in unacceptable memory consumption. Essentially, permutation-based methods rely on the order of nearest pivots, or permutation, of individual data elements[18-22]. Heuristically, two data elements are close if their permutations are similar. Permutation-based methods can be tree-structured[20-22] or table-structured[18-19]. Generally, permutation-based methods suffer from excessive index size, excessive search memory, or low recall with limited memory.

Today, as data volume increases rapidly, it is getting harder and harder to run ANNS of large data sets on commodity machines. Based on in-depth understanding of existing permutation-based methods, we propose the reduced permutation array (RPA) for recall@R ANNS in metric spaces. The major contributions include:

1) We distinguish three critical factors in designing a permutation-based metric-space ANNS index, namely structure, score function, and pivot selection. Then, a survey of existing methods organized by the three factors is presented. It reveals the pros and cons of various design choices, forming a framework to design new methods.

2) RPA grows from the above framework aiming at low memory consumption. For structure, RPA is a 2 dimensional array consisting of only the identities (IDs) of a few nearest pivots for each data elements, resulting in smaller index size. For score function, we show that the metric index (M-index) [21] score function integrating permutation and query-pivot distance fits RPA well and adopt it. For pivot selection, we show that a smarter pivot selection strategy, in contrast to random pivot selection, can achieve the required performance with fewer selected pivots. Furthermore, even a small number of nearest pivots are adequate to distinguish individual data elements.

1　Related　work

1.1　Structure

Common logical structures of indexes are tree or table. Essentially, tree-structured indexes partition data into sub-trees during construction, and try to prune them to reduce the problem size during search. For example, prefix permutation index (PP-index)[20], metric index (M-index) [21], and PPP-codes[22] are tree-structured metric-space ANNS indexes.

Essentially, table-structured indexes organize particular compressed form of each data element as an array during construction, and scan the compressed array to save input/output (I/O) during search. Table structure essentially store the ordered IDs of nearest pivots for each

http://journal.szu.edu.cn

data element. Such information can be logically considered as a matrix, with data elements as the rows and pivots as the columns. Thus, the physical organization of the matrix can be data-major (row-major) or pivot-major (column-major).

Table structure can be further divided into inverted files, suffix trees and permutation arrays. Among them, inverted files, such as metric inverted files (MIF)[19], and suffix trees, such as metric suffix array[23], are both pivot-major, while permutation arrays, such as permutation-based index (P-index)[18] and brief permutation index[24], are data-major.

1.2 Score function

In permutation-based metric-space ANNS, an essential heuristic is that if two data elements' permutation are similar, the two elements are close to each other. Permutations are stored in metric-space permutation-based ANNS indexes. During search, a score function computes, based on data-pivot permutations, query-pivot permutations and/or query-pivot distances, the approximate data-query distance, which is expected to approximate the data-pivot distances or keep the order of them.

Based on the information used, score functions can be categorized into two types. ① Permutation-only, which compare data-pivot permutations with query-pivot permutations to approximate data-query distances. ② Permutation-distance, which additionally utilize query-pivot distances in approximating data-query distances.

Spearman rho and Spearman footrule distance are two typical permutation-only score functions[25]. They are both based on the offsets of each pivot's positions in data-pivot and query-pivot permutations. It has been shown that Spearman rho yields out better accuracy than Spearman footrule distance. Kendall tau is another typical permutation-only score function. It essentially calculates the number of inverted pairs of pivot indices in the pivot permutation of a data element and a query. It has been shown that Kendall tau is slightly better than Spearman rho[25] in accuracy, while more costly in computation. Please be noted that these 3 permutation-only score functions all require that the permutations are based on identical pivots.

Two permutation-distance score functions appear in references [21] and [22]. They both take use of query-pivot distances in addition to data-pivot and query-pivot permutations, and both works on reduced pivots, i. e. the permutations are based on different subsets from a set of pivots. The basic idea is to approximate the data-query distance by the distances from the query to data element's nearest pivots. It has been shown that permutation-distance score functions outperform permutation-only ones[22].

1.3 Pivot selection

Metric spaces do not have coordinates. Consequently, advanced coordinate-oriented mathematical tools are not directly applicable. Pivot selection selects a number of reference points, or pivots, from data, such that the distance vector from a data elements to all pivots can be considered as its coordinate. For pivot selection of permutation-based metric-space ANNS indexes, there are 3 issues to consider, namely objective function and selection algorithm, number of pivots to identify each data element, or length of permutation of each element l, and number of total pivots k.

In permutation-based metric-space ANNS indexes, data elements are considered close if their permutation of nearest pivots are similar. Therefore, data are essentially partitioned by Voronoi diagram with the pivots as the centers of Voronoi cells, and each cell is further partitioned by other pivots, recursively. At the end, data in the same cell have the same permutation. Consequently, if a cell is large, the accuracy will be low. As a result, the locations or selection of pivots are of key importance.

As a fundamental problem of metric-space data processing, pivot selection has received extensive research efforts[6-9, 26-28]. However, most existing permutation-based metric-space ANNS indexes select pivots randomly[18-21]. Some indexes select a lot of pivots[18-21], and PPP-codes[22] builds 4 trees and integrates the search results together.

Some researchers believe that all data elements should use the same set of pivots[18-19, 23-24], i. e. $k = l$. But some other researchs believe that different data elements could be better distinguished by different pivots[20-22], i. e. $k > l$. In the latter case, k pivots are selected for the whole data set. For each data element, only l out of the k pivots are determined to appear in its permutation, called reduced permutation. Therefore, the score function must be applicable to reduced permutations.

http://journal.szu.edu.cn

2　RPA

The aim of RPA is low memory consumption without sacrificing the accuracy. In this section, design strategies of RPA are elaborated from the perspective of structure, score function, and pivot selection. Then, the construction and ANNS algorithms of RPA are presented and analyzed.

2.1　Design strategies

In the following, we explain how RPA saves memory without sacrificing accuracy from the perspectives of structure, score function, and pivot selection.

2.1.1　Structure

It has been shown in reference [29] that as the dimension of data element increases, while the number of data elements fixed, the performance of tree-structured index downgrades to linear scan. Given that ANNS is typically only considered for high-dimensional data, RPA adopts table structure rather than a tree structure.

As discussed above, a table-structure permutation-based metric-space ANNS index stores the IDs of the nearest pivots of every data elements, and the organization can be data-major or pivot-major. It is the search time memory overhead and index size that differ data-major and pivot-major modes. For data-major mode, only a bounded heap of maximal size R holding the candidate results has to be maintained during search. For pivot-major mode, during search, since the distance between a data element to the query has to be approximated by multiple pivots, the number of candidate results could not be safely discarded could be rather large or even close to the size of data set when only part of the pivots are scanned, resulting in huge memory overhead. As to index size, let n be the data set size, k be the total number of pivots, and l be the number of the nearest pivot stored for each data element, while $l < k \ll n$. Thus, the total number of IDs to be stored in the index for both modes is nl. Each ID takes lb$k$ bits for data-major mode and lb n bits for pivot-major mode. That is, a data-major index takes $nl\,\mathrm{lb}\,k$ bits, less than the $nl\,\mathrm{lb}\,n$ space taken by a pivot-major index. Please be noted that, since $k \ll n$ in general, a small amount of memory to store the pivots is ignored. Therefore, RPA deploys data-major mode, forming an array of permutations, i. e. IDs of nearest pivots, of every data elements, consecutively Fig. 1.

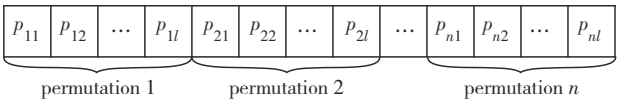


Fig. 1　Structure of RPA. (The subscript i in p_{ij} represents the serial number of the data element, and the subscript j ranges from 1 to l. The smaller the j, the closer the support point is to the data element i.)

图 1　RPA 结构(其中，p_{ij} 中的下标 i 表示数据元素的序号，下标 j 范围从 1 到 l，j 越小代表支撑点距离数据元素 i 越近)

2.1.2　Score function

Permutation-only score functions consider only the positions of pivots in permutations, and require identical pivot set for two permutations. Intuitively, permutation-only score functions are more qualitative than quantitative. On the contrary, permutation-distance score functions take query-pivot distances into consideration, and are applicable to permutations with different pivots. Intuitively, permutation-distance score functions are more quantitative.

The PPP-codes score function computes a weighted mean of the distances from the query to the nearest pivots of data elements, while farther pivots with smaller weights[22]. The M-index score function sums the difference, called penalty, of the distances from the query to its nearest pivots and to data element′s nearest pivots[21]. We believe that the M-index score function is more accurate, and preliminary experimental results also confirm that it works better with RPA.

2.1.3　Pivot selection

Good pivots lead to accurate approximation of the query-data distances, or at least the order of them. We believe that a small number of carefully selected pivots could perform as well as a lot of randomly sampled pivots, without building multiple indexes.

A widely accepted idea is that good pivots are usually from corner or border of data[26-28, 30-31]. To this end, RPA deploys the farthest-first traversal (FFT)[32-34] algorithm to select pivots. The first pivot is the farthest data element to the first element. Then, FFT iteratively selects the data element whose minimal distance to the current set of pivots is the maximum among others. As a k-center clustering algorithm, FFT takes linear time and space, and guarantees the largest cluster diameter is at most twice of the optimum. FFT is known to select pivots from the

http://journal.szu.edu.cn

border of data, and is widely deployed for pivot selection[33, 35-37].

Many previous work propose that the number of pivots should be close to the intrinsic dimension of data[26, 31]. The idea behind is that the distances from a data element to such number of pivots are enough to distinguish that element. Similarly, it is reasonable to believe a length close to the intrinsic dimension of data is enough for the permutation to accurately distinguish a data element, given that the pivots were properly selected. That is, the number of pivots in each data element′s permutation l, should be close to the intrinsic dimension of data. Ways to estimate the intrinsic dimension of data have been extensively discussed in prior research[26-27, 38], and this topic is beyond the scope of this paper.

RPA adopts the idea that distinct data elements can be more accurately distinguished by different pivots[20-22]. Therefore, RPA uses a larger value of k than l to make the reduced pivots permutation of each data point as different as possible.

2.2 Algorithms and analysis

Based on the design strategies discussed above, the construction and approximate nearest neighbor (ANN) search algorithms of RPA are presented and analyzed in the following.

2.2.1 Construction

According to the structure of RPA, its construction algorithm is rather straightforward (Algorithm 1 in Fig. 2).

First, k pivots are selected via the FFT algorithm (line 2). Then, implemented by a bounded heap of maximal size l. For each data element, its distances to all pivots are computed (line 6), the l nearest pivots are distinguished (line 7). Then, every data item′s pivot permutation (pivot IDs) are appended to the index array (line 9).

Obviously, the time and space complexities of the construction are both $O(n)$. The RPA data structure returned takes $nl \operatorname{lb} k$ space.

2.2.2 ANN search

The recall@R ANNS algorithm (Algorithm 2 in Fig. 3) is as simple as a linear scan.

The distances from the query to the pivots are computed (line 4). The permutation of the query is generated (line 5), implemented by a bounded heap of maximal size l. Then, the index array is scanned linearly (lines 7 to 10). For each data element, its score or approximate dis-

input: data set U, pivot set P, distance function d, total number of pivot k, length of each permutation l
output: the array of reduced permutations of all data elements RPA

```
1  RPA ← { }
2  P ← FFT(k) // select k pivots via the FFT algorithm
3  for each x ∈ U do
4      H ← an empty bounded heap of maximal size l
5      for each p_i ∈ P do
6          dist ← d(p_i, x) // calculate the distance from x to pivot p_i
7          H. insert(i, dist) // insert (i, dist) into bounded heap, ordered by dist
8      end for
9      RPA. add(Encode(H)) // append x′s permutation in bit coding to RPA
10 end for
11 return RPA
```

Fig. 2 RPA construction (Algorithm 1).
图2 RPA构建的伪代码(算法1)

Input: reduced permutation array RPA, pivot set P, distance function d, score function Score, query q, length of each permutation l, result set size R
Output: the search results

```
1  RHeap ← an empty bounded heap of maximal size R to keep the result
2  QHeap ← an empty bounded heap of maximal size l to keep q's permutation
3  for each p_i ∈ P do
4      dist[i] ← d(p_i, q)
5      QHeap. insert(i, dist[i]) // insert (i, dist[i]) into the result heap, ordered by the distance from q to p_i
6  end for
7  for each RPA_i ∈ RPA do // scan the reduced permutation array
8      s = Score(RPA_i dist, QHeap) // compute the score function
9      RHeap. insert(i, s) // insert (i, s) into the result heap, ordered by score
10 end for
11 return RHeap
```

Fig. 3 RPA search (Algorithm 2).
图3 RPA搜索算法伪代码(算法2)

http://journal.szu.edu.cn

tance to the query is computed by the same score function as used in M-index[21], based on the permutations of the data element and the query, and the query-pivot distances (line 8). The R results with the smallest scores are distinguished, implemented by a bounded heap of maximal size R (line 9). Obviously, the time complexity of this ANN search algorithm is $O(n)$, and the space overhead is only the space to store at most R IDs of search results and their score or approximate distance to the query.

Due to the simplicity of the structure and the two algorithms above, RPA is able to easily support dynamic operations such as insertion, deletion, and modification. It is also rather easy to be implemented in parallel, just by partitioning the data, and the parallel efficiency is almost guaranteed to be high. Certainly, periodically re-selecting the pivots could be beneficial.

3 Experimental evaluation

In the experiments, we focus on four key aspects. Firstly, we validate the influence of score functions and pivot selection algorithms on performance. Secondly, the length of the permutation directly affects memory efficiency, we will delve into the detailed determination of the values of k and l. Finally, we will conduct comparative experiments between RPA and permutation-based index.

3.1 Data sets and experiment set up

Three data sets from reference [26] are used, i. e. 20 dimensional random vectors (UV20), feature vectors of images (Image), and DNA 18-mers (DNA18). Table 1 summarizes these data sets, and more details can be found in reference [26]. To save experimental time, the data set sizes are limited. Due to the simplicity of the structure and algorithms of RPA, we believe similar conclusions can be drawn from larger data sets. For each data set, a random sample is selected as the query set. The value of R in calculating the recall is 100 for all the experiments.

Table 1　Experimental data sets[26]

表 1　实验数据集[26]

data set name	$10^{-4}\times$ data set size	$10^{-4}\times$ query set size	distance metric	dimension
UV20	10	1	L^2 norm	20
DNA18	50	1	Hamming distance	18
image	1	1	L norm	66

3.2 Comparison of M-index and PPP-codes score functions

Since $k \neq l$ for RPA, only permutation-distance score functions can be adopted. Currently, there are only two score functions available: the M-index score function and the PPP-code score function. Consequently, we conduct experiments on random vector data sets to compare the performance of RPA using the these two score functions. We set $k = 2l$, and tested various values of l. Fig. 4 is the M-index and PPP-codes score functions on UV20. It shows that the M-index score function outperforms the PPP-codes one.

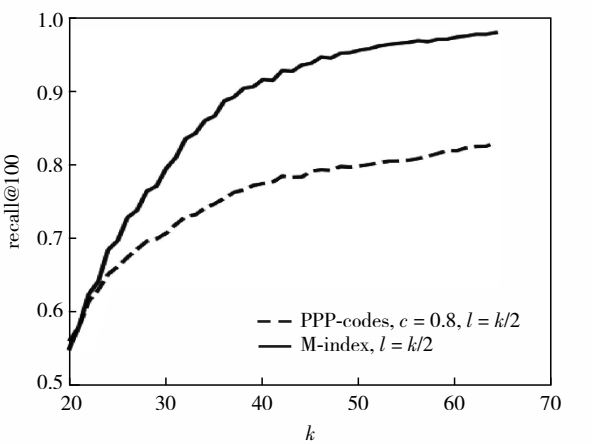


Fig. 4　M-index and PPP-codes score functions on dataset UV20.

图 4　M-index 与 PPP-codes 打分函数在 UV20 数据集上的性能比较

3.3 Effect of different pivot selection algorithms

This part of experiments are performed on the UV20 dataset. The value of k is 21. Three pivot selection algorithms are considered, namely random selection, incremental selection[28] and FFT. Figure 5 shows that FFT achieves the highest recall@100 with less number of pivots, compared with the other two algorithms. Therefore, RPA adopts FFT pivot selection algorithm to reduce memory consumption.

3.4 Determining k and l

To explore how k and l affect the index performance, recall in particular, we experiment with various values of them on 3 data sets. The results are plotted in Fig. 6.

The observation Fig. 6 shows:

1) For all the cases, given fixed k, as l increases, recall increases rapidly at the beginning, then slows down, and approaches the final value slowly in the last stage.

http://journal.szu.edu.cn

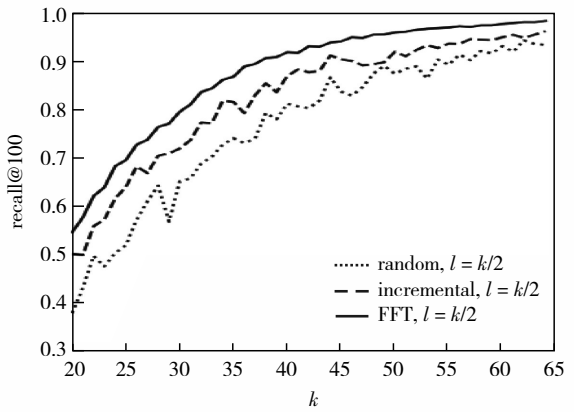


Fig. 5 Comparison of the three pivot selection methods.

图5 三种支撑点选择的比较

Proper values of l could be 30 for UV20, 35 for DNA, and 10 for Images.

2) As k increases, the overall recall increases. There exists a critical threshold beyond which the incremental increase in k has a negligible effect on improving the recall. Proper values of k could be 128 for UV20, 256 for DNA18, and 64 for Image.

Given a data set, to determine k and l, we suggest to work on a small sample, first set $k = l$ and find a proper value of k based on the recall required by the application, and then fix k and reduce l to find a proper value for it.

3.5 Comparing the memory-consumption of RPA with existing works

As discussed in section 1, tree-structured permutation-based metric-space ANN indexes, such as PP-index[20], M-index[21], and PPP-codes[22] are suffered from the high dimensionality of data. Pivot-major table-structured permutation-based metric-space ANN indexes, such as MIF[19] and metric suffix array[23], are suffered from huge memory overhead during search. To save space, we leave the comparisons with methods mentioned above in the future work.

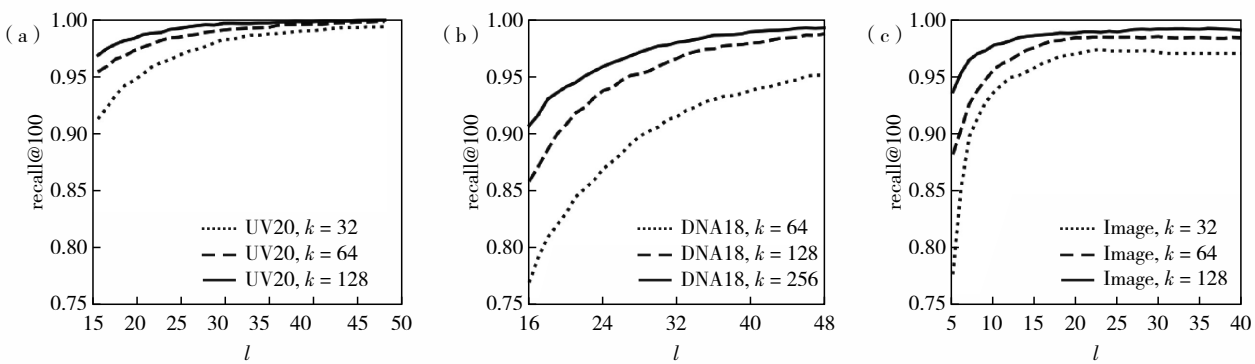


Fig. 6 Performance of with different k and l on dataset of (a) UV20, (b) DNA18, and (c) Image.

图6 不同k和l组合的RPA搜索性能在数据集(a)UV20、(b)DNA18和(c)Image上的结果

In order to intuitively reflect the overall performance advantages of our approach, RPA is compared with an existing data-major table-structured permutation-based metric-space ANN index, P-index[18]. RPA differs from P-index in reduced permutation, score function from M-index, and pivot selection by FFT.

Experimental results are listed in Table 2. For each data set, 3 recall values of different levels are picked, the corresponding k are collected, and the size of P-index are computed by $nk \operatorname{lb} k$. For a fair comparison, RPA employs an equivalent value of k as P-index, and then reduces l to achieve slightly higher recalls compared to those of the P-index. The values of k and l are collected, and the size of RPA are computed by $nl \operatorname{lb} k$. Please be noted that, since $k \ll n$ in general, a small amount of memory to store the pivots is ignored. The ratio of the index size of P-index over that of RPA is listed. It can be seen that P-index generally takes 3 times, and up to 4 times, index space as that of RPA. The search speed and memory overhead of both methods are similar, and are thus not listed.

Conclusions and future work

This paper targets at the excessive memory consumption problem of permutation-based recall@R ANNS indexes in metric spaces. We first distinguish 3 key design factors, namely structure, score function and pivot selection. According the 3 factors, existing works are analyzed, and pros and cons are discovered. Then,

http://journal.szu.edu.cn

Table 2　Compare the index sizes of P-index and RPA under the same recall@100

表2　P-index 和 RPA 在相同 recall@100 召回率情况下的索引

dataset	P-index			RPA				index size
	recall@100/%	k	index size/kbits	recall@100/%	k	l	index size/kbits	ratio[1)]
UV20	89.93	46	27 600	90.14	46	17	10 200	2.71
	95.03	56	33 600	95.08	56	22	13 200	2.55
	99.01	83	58 100	99.06	83	36	25 200	2.31
DNA18	90.47	72	252 000	90.85	72	28	98 000	2.57
	95.31	96	336 000	95.46	96	34	126 000	2.82
	99.04	192	768 000	99.11	192	50	200 000	3.84
Image	85.41	18	900	86.13	18	10	500	1.08
	90.08	29	1 450	91.20	29	9	450	3.22
	95.12	42	2 520	95.54	42	11	660	3.82

1) The index size ratio is the ratio of P-index index size to RPA index size.

based on this in-depth survey, RPA is proposed. RPA integrates data-major tree structure, M-index score function, reduced permutation, and pivot selection by FFT. Preliminary experimental results show that RPA is up to 4 times memory-efficient than P-index.

In the future work, RPA will be experimentally compared with more existing works such as M-index and PPP-codes. More data sets, such as the Similarity search and applications (SISAP) data sets, should be included. Better score functions, possibly integration of the M-index and PPP-codes ones, are of interest. More pivot selection objective functions and algorithms should be studied together with RPA. Last but not least, comparative study with graph-based ANNS indexes is an interesting direction.

基金项目：国家自然科学基金资助项目(62072311)
作者简介：江润本(jiangrunben2021@email.szu.edu.cn)，深圳大学硕士研究生．研究方向：通用搜索技术．
引　　文：江润本，陈家颖，毛睿．RPA：一种内存高效的度量空间 recall@R 近似最近邻搜索索引[J]．深圳大学学报理工版，2023，40(6)：640-648.(英文版)

参考文献 / References:

[1] COVER T, HART P. Nearest neighbor pattern classification [J]. IEEE Transactions on Information Theory, 1967, 13(1): 21-27.

[2] ARYA S, MOUNT D M, NETANYAHU N S, et al. An optimal algorithm for approximate nearest neighbor searching fixed dimensions [J]. Journal of the ACM, 1998, 45(6): 891-923.

[3] MUJA M, LOWE D G. Fast approximate nearest neighbors with automatic algorithm configuration [C]// Proceedings of the 4th International Conference on Computer Vision Theory and Applications. Lisboa, Portugal: VISAPP, 2009: 331-340.

[4] JÉGOU H, DOUZE M, SCHMID C. Product quantization for nearest neighbor search [J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2011, 33(1): 117-128.

[5] KELLEY J L. General topology [M]. [S. l.]: Courier Dover Publications, 2017.

[6] UHLMANN J K. Satisfying general proximity / similarity queries with metric trees [J]. Information Processing Letters, 1991, 40(4): 175-179.

[7] CHÁVEZ E, NAVARRO G, BAEZA-YATES R, et al. Searching in metric spaces [J]. ACM Computing Surveys, 2001, 33(3): 273-321.

[8] ZEZULA P, AMATO G, DOHNAL V, et al. Similarity search the metric space approach [M]. Boston: Springer US, 2006.

[9] SAMET H. Foundations of multidimensional and metric data structures [M]. Boston: Morgan Kaufmann, 2006.

[10] HJALTASON G R, SAMET H. Index-driven similarity search in metric spaces (survey article) [J]. ACM Transactions on Database Systems, 2003, 28(4): 517-580.

[11] AUMÜLLER M, BERNHARDSSON E, FAITHFULL A. ANN-benchmarks: a benchmarking tool for approximate nearest neighbor algorithms [J]. Information Systems, 2020, 87: 101374.

[12] MALKOV Y, PONOMARENKO A, LOGVINOV A, et al. Approximate nearest neighbor algorithm based on navigable small world graphs [J]. Information Systems, 2014, 45: 61-68.

[13] FU Cong, XIANG Chao, WANG Changxu, et al. Fast approximate nearest neighbor search with the navigating spreading-out graph [J]. Proceedings of the VLDB Endowment, 2019, 12(5): 461-474.

[14] HARWOOD B, DRUMMOND T. FANNG: fast approximate nearest neighbour graphs [C]// The IEEE Conference

http://journal.szu.edu.cn

on Computer Vision and Pattern Recognition. Piscataway, USA: IEEE, 2016: 5713-5722.
[15] DONG Wei, MOSES C, LI Kai. Efficient *k*-nearest neighbor graph construction for generic similarity measures [C]// Proceedings of the 20th International Conference on World Wide Web. New York, USA: Association for Computing Machinery, 2011: 577-586.
[16] WANG Mengzhao, XU Xiaoliang, YUE Qiang, et al. A comprehensive survey and experimental comparison of graph-based approximate nearest neighbor search [J]. Proceedings of the VLDB Endowment, 2021, 14(11): 1964-1978.
[17] YANG Jianye, YAO Wu, ZHANG Wenjie. Keyword search on large graphs: a survey [J]. Data Science and Engineering, 2021, 6(2): 142-162.
[18] CHAVEZ E, FIGUEROA K, NAVARRO G. Effective proximity retrieval by ordering permutations [J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2008, 30(9): 1647-1658.
[19] AMATO G, GENNARO C, SAVINO P. MI-file: using inverted files for scalable approximate similarity search [J]. Multimedia Tools and Applications, 2014, 71(3): 1333-1362.
[20] ESULI A. PP-index: using permutation prefixes for efficient and scalable approximate similarity search [C]// Proceedings of the 18th Italian Symposium on Advanced Database Systems. Boston, USA: LSDS-IR Workshop, 2010.
[21] NOVAK D, BATKO M, ZEZULA P. Metric index: an efficient and scalable solution for precise and approximate similarity search [J]. Information Systems, 2011, 36(4): 721-733.
[22] NOVAK D, ZEZULA P. PPP-codes for large-scale similarity searching [M]// HAMEURLAIN A, KÜNG J, WAGNER R. Transactions on Large-Scale Data- and Knowledge-Centered Systems XXIV: Volume 9510. Berlin: Springer-Verlag, 2016: 61-87.
[23] MOHAMED H, MARCHAND-MAILLET S. Metric suffix array for large-scale similarity search [C]// ACM WSDM 2013 Workshop on Large Scale and Distributed Systems for Information Retrieval. Rome, Italy: ACM, 2013: 17780703.
[24] TÉLLEZ E S, CHÁVEZ E, CAMARENA-IBARROLA A. A brief index for proximity searching [C]// Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications. Berlin: Springer Berlin Heidelberg, 2009: 529-536.
[25] FAGIN R, KUMAR R, SIVAKUMAR D. Comparing top *k* lists [J]. SIAM Journal on Discrete Mathematics, 2003, 17 (1): 134-160.
[26] MAO Rui, MIRANKER W L, MIRANKER D P. Pivot selection: dimension reduction for distance-based indexing [J]. Journal of Discrete Algorithms, 2012, 13: 32-46.
[27] MAO Rui, ZHANG Peihan, LI Xingliang, et al. Pivot selection for metric-space indexing [J]. International Journal of Machine Learning and Cybernetics, 2016, 7(2): 311-323.
[28] BUSTOS B, NAVARRO G, CHÁVEZ E. Pivot selection techniques for proximity searching in metric spaces [J]. Pattern Recognition Letters, 2003, 24(14): 2357-2366.
[29] WEBER R, SCHEK H J, BLOTT S. A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces [C]// Proceedings of the 24rd International Conference on Very Large Data Bases. San Francisco, USA: Morgan Kaufmann Publishers Inc, 1998: 194-205.
[30] YIANILOS P N. Data structures and algorithms for nearest neighbor search in general metric spaces [C]// Proceedings of the 4th Annual ACM-SIAM Symposium on Discrete Algorithms. [S. l]: Society for Industrial and Applied Mathematics, 1993: 311-321.
[31] TRAINA C, FILHO R F S, TRAINA A J M, et al. The omni-family of all-purpose access methods: a simple and effective way to make similarity search more efficient [J]. The VLDB Journal, 2007, 16(4): 483-505.
[32] GONZALEZ T F. Clustering to minimize the maximum intercluster distance [J]. Theoretical Computer Science, 1985, 38: 293-306.
[33] BERMAN A, SHAPIRO L G. Selecting good keys for triangle-inequality-based pruning algorithms [C]// Proceedings of the IEEE International Workshop on Content-based Access of Image and Video Database. Piscataway, USA: IEEE, 1998: 12-19.
[34] HOCHBAUM D S, SHMOYS D B. A best possible heuristic for the *k*-center problem [J]. Mathematics of Operations Research, 1985, 10(2): 180-184.
[35] VLEUGELS J, VELTKAMP R C. Efficient image retrieval through vantage objects [J]. Pattern Recognition, 2002, 35 (1): 69-80.
[36] MAO Rui, XU Weijia, SINGH N, et al. An assessment of a metric space database index to support sequence homology [J]. International Journal on Artificial Intelligence Tools, 2005, 14(5): 867-885.
[37] HENNIG C, LATECKI L J. The choice of vantage objects for image retrieval [J]. Pattern Recognition, 2003, 36(9): 2187-2196.
[38] CLARKSON K L. Nearest-neighbor searching and metric space dimensions [M]// SHAKHNAROVICH G, DARRELL T, INDYK P. Nearest-Neighbor Methods in Learning and Vision: Theory and Practice. Cambridge, USA: MIT Press, 2006: 15-59.

【中文责编：英子；英文责编：木柯】

http://journal.szu.edu.cn