Effective Use of Space for Pivot-Based Metric Indexing Structures

Cengiz Celik*
Department of Computer Engineering
Bilkent University
Ankara, Turkey
ccelik@cs.bilkent.edu.tr

Abstract

Among the metric space indexing methods, AESA is known to produce the lowest query costs in terms of the number of distance computations. However, its quadratic construction cost and space consumption makes it infeasible for large datasets. There have been some work on reducing the space requirements of AESA. Instead of keeping all the distances between objects, LAESA appoints a subset of the database as pivots, keeping only the distances between objects and pivots. Kvp uses the idea of prioritizing the pivots based on their distances to objects, only keeping pivot distances that it evaluates as promising. FQA discretizes the distances using a fixed amount of bits per distance instead of using system's floating point types. Varying the number of bits to produce a performance-space trade-off was also studied in Kvp. Recently, BAESA has been proposed based on the same idea, but using different distance ranges for each pivot. The t-spanner based indexing structure compacts the distance matrix by introducing an approximation factor that makes the pivots less effective.

In this work, we show that the Kvp prioritization is oriented toward symmetric distance distributions. We offer a new method that evaluates the effectiveness of pivots in a better fashion by making use of the overall distance distribution. We also simulate the performance of our method combined with distance discretization. Our results show that our approach is able to offer very good space-performance trade-offs compared to AESA and tree-based methods.

1. Introduction

One of the ways to categorize the existing indexing methods in metric spaces is based on their data organization. Tree structures have a top-down approach, building the index in a hierarchical manner. The alternative bottom-up organization is represented by the family of flat, global pivot-based methods. These structures usually compute the distance of every object to the pivots. The amount of space consumption has not been a major issue for tree structures since they use very limited space such as the *Vp-tree* [17], *GH-tree* [13] and the *Mvp-tree* [1]; or reside in secondary memory such as the *M-tree* [9], *Slim-tree* [5], *DF-tree* [4] and the *DBM-tree* [16].

The *AESA* [14] takes a different approach by computing and storing all the distances between objects. It offers very good query performance in terms of the number of distance computations, but its quadratic setup time and space usage makes it infeasible for many applications. *LAESA* [11] was introduced to decrease the space and construction costs of AESA. LAESA uses a subset of the database as pivots, so that the rest of the database objects only compute their distances to these pivots. Although ways to select the set of pivots to optimize query performance has been proposed [3], these are tailored toward improving the distance relationships among pivots only.

A typical flat structure uses more memory than a tree counterpart while offering better query performance. These structures have also usually been implemented in main memory. Reducing the space overhead of global pivot-based structures not only decreases the query processing times due to the less data to be processed, but also makes it feasible to store more objects in memory. We believe that with the positive trends in memory capacities and prices, even very large databases should have the potential to be stored or at least indexed completely in main memory.

One of the ways of reducing the storage requirements of a global pivot-based method is to store the distances in lesser precision, as done in *FQA* [7] and *BAESA* [10]. This method is described as *range coarsening* in [8]. Well-known tree structures like the Vp-tree [17], GNAT [2] and M-tree [9] also use the same concept to partition their nodes. Another way of reducing space is to store only a subset of the distances, defined as *scope coarsening* in [8]. Tree

*This work is supported by TUBITAK Career Grant 106E130.

0-7695-3101-6/08 $25.00 © 2008 IEEE
DOI 10.1109/SISAP.2008.22

structures attempt to cluster relevant objects together; as they descend down the tree, they pick pivots only from the local population. This way, depending on the quality of the tree's clustering, pivots govern over their close neighbors, upon which they are more effective. The *Kvp* structure [6] is built over the observation that pivots are more effective on close and far objects. The structure only stores these promising distances and reduces the space requirements to any desired level at the cost of query performance.

The *t-spanners structure* [12] approximates the original distance matrix by introducing an error rate that can be bounded. The setup time for the t-spanners is even worse than the AESA, limiting its use for only small cardinalities, but its query performance is close to AESA while consuming less memory. For some specific datasets, it has been reported to use 4% of the space required by AESA, resulting in an extra 9% query cost overhead. In vector spaces the performance has been reported to be worse, producing at best twice the cost of AESA using about 15% of its space. Taking into consideration the huge performance gap between AESA and tree-based structures, this is a very good trade-off. The quality and setup time of t-spanners is also sensitive to the intrinsic dimension of data. It has been reported that the construction time increases to impractical levels for dimensions higher than 24.

BAESA [10] is another approximation of AESA. For each object, it defines different distance ranges to be used for the discretization of its distances to other objects. Using 2^k ranges, BAESA needs to store an extra 2^k distances but only k bits per object per distance. For example, if the built-in floating point type uses 64 bits, their best results are reported using $k = 4$, hence using $16 \cdot 64 + (n/2) \cdot 4$ bits per object compared to $64 \cdot (n/2)$ bits per object used by AESA. At its best, it was reported to approximately produce twice the cost of AESA using slightly more than $1/8$ of its space. BAESA was also reported to provide better performance using same amount of space after 12 dimensions in uniform vector space.

In this paper we will improve the prioritization scheme of Kvp, and study the performance of our methods in deeper detail. We will propose structures that approximate AESA, as well as structures that have linear setup cost and are more practical for large databases.

2. Distribution Sensitive Pivot Prioritization

Given a space threshold, the Kvp uses half of its space for close pivots, and the other for far pivots. Kvp also has a parameter that controls the ratios of these two groups, but there is no fixed guidelines for optimizing this parameter. Ideally, an indexing structure should adjust itself to the dataset it is using. Also, one can come up with distributions for which far and close pivots are the worst choices. In order to make the discussion more concrete, we start by describing the effectiveness of a pivot relative to a database object. We define the *pivot efficiency* with respect to a query radius r as:

$$\mathit{Eff}(p, r) = \text{prob. that an object will be eliminated by } p \text{ for a query of radius } r$$

Given $F()$, the cumulative probability distribution function, assuming that query objects are drawn from the same distribution as database objects, we can approximate the efficiency of p over a database object o at a distance D_{po} using the following equation.

$$\mathit{Eff}(p, o, r) \approx F(D_{po} - r) + (1 - F(D_{po} + r)) + F(r - D_{po}) \quad (1)$$

where the first term in summation represents the probability that o is sufficiently far from the pivot and the query object, the second term represents the case when the query object is far from the pivot, and the third term represents the rare case that both the query object and o are close enough to the pivot to prove that o is in the query range.

We have run a series of experiments in which we computed the pivot efficiencies for all possible distance values. Our results are summarized in Fig. 1. One of the important observations here is that there are huge differences in the degrees of efficiencies of pivots. It certainly does not make sense to store all the distances since most of them are pretty useless. We also see that the query radius has an effect on the pivot efficiencies, but preserves their relative performances at similar levels. This tells us that prioritizing the pivots for any sensible query radius value will be suitable for other query ranges as well.

We see that symmetrical distance distributions like the uniform, Laplace and Gaussian supports the Kvp methodology. The far and close distances have similar efficiency values. However, the clustered distributions show that close distances are more valuable than the far distances.

Under the light of the above observations, we propose a new global pivot-based indexing scheme. At the construction time, we will sample some arbitrary distances between database objects to get an approximation of the overall distance distribution. For each object, after computing the distances to pivots, we evaluate each pivot distance based on Equation 1. We only store the most promising pivot distances depending on the space limit.

Note that computing the efficiency of a single pivot distance involves 3 integrals. In our implementation we divided the distances into fixed-width ranges and computed a single probability distribution value for the whole bin. This approximation reduces the evaluation step to 3 table lookups. Certainly, using a fixed query radius, we can also summarize the pivot efficiencies in a single table. As a result, we argue that the pivot priorization step involves only negligible performance overhead.

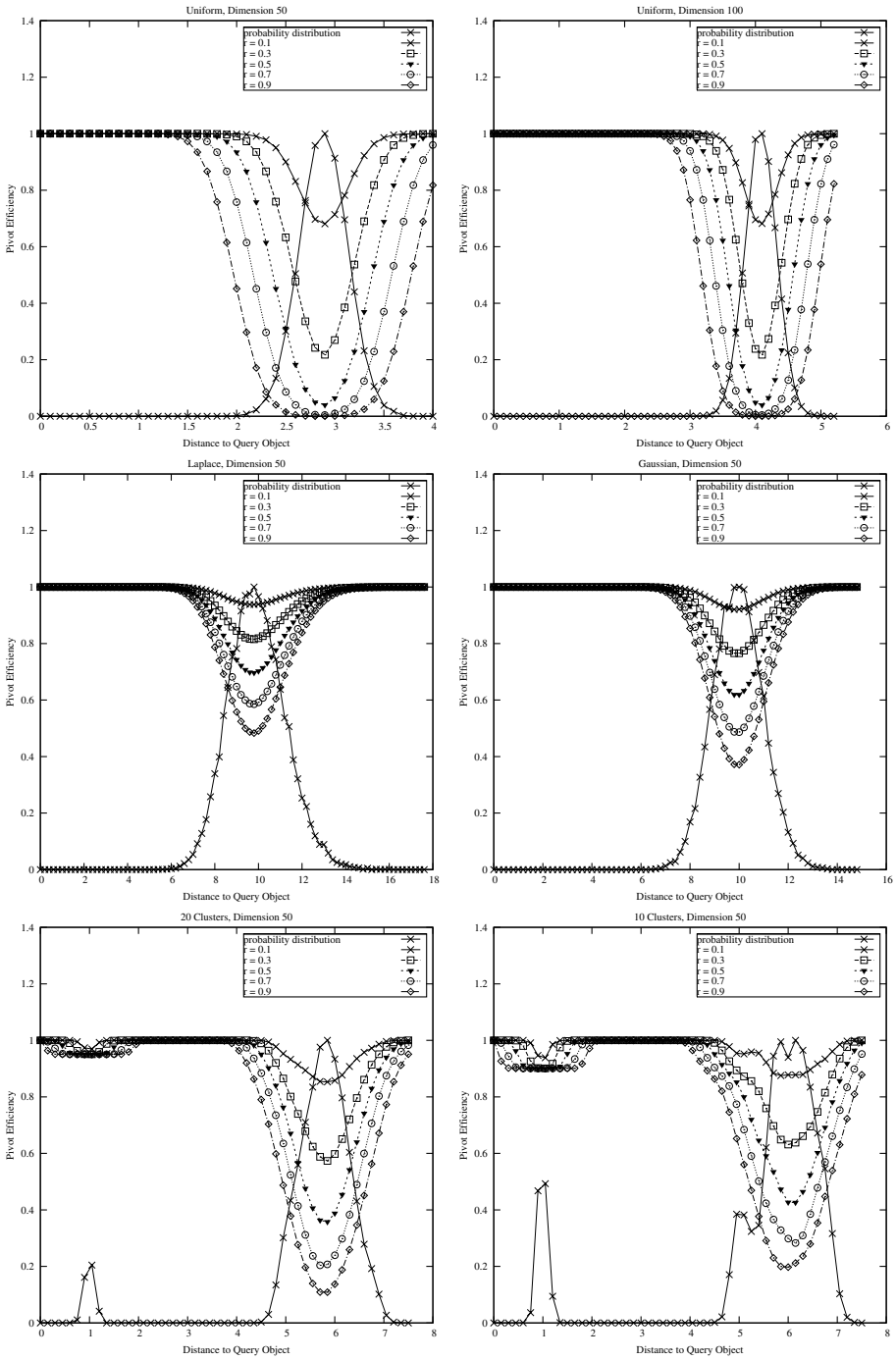

Figure 1. Pivot Efficiencie values for a variety of synthetic vector distributions. The probability distribution is scaled so that the maximum value is translated to 1. Clustered Distributions are generated using a standard deviation of 0.1 within the clusters

We first apply our scheme to AESA [14]. Note that one of the crucial points for the query performance of AESA is to find the most promising pivot to be processed next. The original AESA evaluates each object by how close it seems to be to the query object. Each processed pivot provides a lower bound for the distance between a database object and the query object. AESA uses the sum of these lower bounds as a criteria for the closeness to the query object. The object with the minimum of the sum of the projected lower bounds is chosen next.

Note that this method is not feasible for our scheme, since we do not have distances to all possible pivots. Although using the average of the available lower bounds is also an option, we chose to use the method defined in [15] which seems to offer better performance in high dimensions. In this method, the object with the minimum of the lower bounds is selected to be processed next. We call this variation *AESADD*. Application of the Kvp prioritization to AESA will be called *AESAKvp*.

We also propose a new structure similar to Kvp and LAESA, having a fixed set of pivots. We prioritize the pivot distances similar to Kvp, keeping only promising pivot distances to preserve space and to avoid the extra CPU overhead of sifting through the whole set of distances. We call this structure *KvpDD*. Unlike AESA variants, KvpDD does not have a quadratic construction cost. We think that this scheme is more suitable for scaling to large cardinalities.

Another family of our proposals can be generated by using a limited amount of bits per distance instead of the system's floating point type. We have not actually implemented these variations, but it is very simple to simulate the query performance of these structures. Given b, the number of bits to use for each distance, the most straight-forward way of discretizing the distances is to divide them into 2^b fixed-width ranges. Given that maximum possible distance in a particular distribution is D_{max}, each of these ranges will have a width of $D_{max}/2^b$. This means that, assuming that the built-in floating point type has infinite precision, for each query radius r we will have an error of at most $D_{max}/2^b$. By extending the query radius to $r + D_{max}/2^b$ we prevent the possibility that an object is falsely eliminated because of the approximation. Biasing toward not elimination does not introduce errors because we compute the real distance of each un-eliminated object at the last phase. In our experiments, the discretized version of AESADD using b bits will be denoted as "AESADD b bits", and similarly discretized AESAKvp will be denoted as "AESAKvp b bits".

3. Performance

In this section we will evaluate the performance of our prioritization scheme. We will take the performance of the AESA as our baseline, and compare the space-query cost trade-off of the other structures relative to AESA.

In some of our experiments we have standardized the selection of query radius by the concept of RRM, short for "Radius Relative to Mean". In a lot data distributions the determination of the maximum or minimum possible distance varies highly based on the sample at hand, whereas the mean of the distances is far more stable. An RRM value of r corresponds to a query radius of $r \cdot D_m$, where D_m is the mean of the distances between objects and is determined using sampling.

The *space ratio* value represents the ratio of the distances of objects we keep compared to AESA. For example, when the space ratio is 0.3, we keep the distances of each object to the most promising $0.3 \cdot (n-1)/2$ other objects. Note that AESA keeps on average $(n-1)/2$ distances per object. This is the same criteria the t-spanner had used, and gives a picture independent of the word size. We would like to point out that the actual implementation needs to pay some extra space cost to compress the sparse distance matrix. The simplest implementation would be to keep the ids of the objects that the distances belong to. For a word size of 32, using an extra 16 bits per distance, the simple scheme would bring an extra 50% space cost.

We have observed that the choice of the query radius has an impact on the relative performance of other structures. As the query radius drops, it becomes more difficult to match the query performance of AESA. What is the typical RRM value used in real databases? We think that the answer depends on the particular application. Some data have very small clusters that do not substantially affect the overall distance distribution and thus the query performance; but executing a small query radius would be enough to return all the relevant objects in the same cluster. An example is the database of English words. As a result, we will try to give a balanced view using different query radius values. Increasing query ranges to impractical levels can make any AESA variation look very good on paper, since the query cost of AESA will be close to the size of the database, and there will be less room for AESA to improve on other structures.

We start by looking at the query performance for uniformly distributed random vectors in 50 dimensions using the Euclidean distance. We compare the outcome for RRM=0.2 and RRM=0.3 in Fig. 2. We observe that the performance of the Kvp and DD variants are very close. Based on the results published in [6] we decided to apply 10 bit discretization to our structures which seem to yield a good overall trade-off. Using 4 bits drops the space requirement considerably, but as the results suggest, the query performances drop too much to offer a competitive trade-off. We also included a version of AESA that stores all the distances but using 4 bits per distance. Note that this is shown as a single point in our plot. This represents a solution very sim-

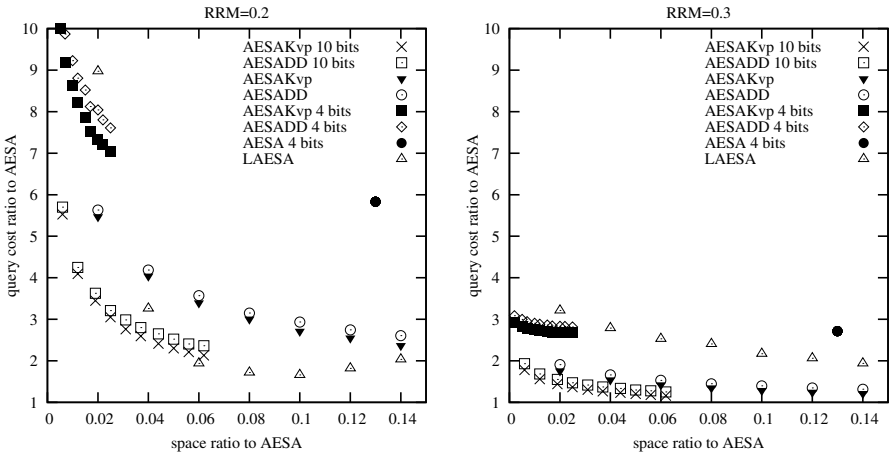

Figure 2. Comparison of AESA variants for 2000 uniformly distributed random vectors in 50 dimensions

ilar to BAESA QBR (Quantiles by Radius [10]). By having a global discretization scheme, we do not need to store an extra of 16 words of distance per object, at the cost of having looser distance ranges, hence possibly reduced query performance. Our results suggest that the trade-off point achieved by 4 bits discretization of AESA is not competitive.

We also compare our schemes to LAESA. The t-spanner structure is reported to be consistently beaten by LAESA that uses the same number of distances for random vector spaces. We think that the main reason behind this is the utilization rate of the distances stored in each structure. In any pivot based structure, under normal conditions, only a portion of the database is used as pivots, and the rest is eliminated based on these pivots. In LAESA, these pivots are fixed. In AESA, the set of pivots we actually use will depend on the pivots' distances to the query object. For every query, we will end up using a different set of pivots. Some of the distances that we have stored will be under-utilized because they will involve eliminated objects. Regardless of the inherent disadvantage of AESA variants that strive to store all possible distance combinations, we see that AESADD and AESAKvp usually beats LAESA that stores the same number of distances.

Going to RRM=0.3, we see that the AESA approximations are able to provide much better query performance for the same rates of space consumption. Their relative comparison stays at similar levels, except for LAESA. we see that more difficult queries favor AESADD and AESAKvp over LAESA.

We have not plotted in our figures, but when we run our queries for RRM=0.4, AESAKvp can get as close as 0.7% of the cost of AESA using 20% of its space. Using 10 bits, it can also provide 2% query cost overhead using just 4.4% of the space. For RRM=0.5 and using 10 bits, AESAKvp uses 2% of the space of AESA, having only 2% more query cost.

Figure 3 provides more results for a comparison of our scheme with Kvp. With the symmetric distributions of Gaussian and Laplace we observe that AESADD and AESAKvp variants are very close in performance. AESADD shows a superior performance when used on clustered distributions. Our scheme outperforms the Kvp variants up to a factor of 10. Here we also see that 4 bit variants offer a better trade-off. We think that this is because the queries have become too easy for the extra bits to make any difference.

As we have argued before, AESA variants are not practical for large cardinalities. We have also observed that AESADD loses it effectiveness as database grows. Fig. 4 shows that AESADD performs relatively worse than the case for 2000 objects as demonstrated earlier in Fig. 2. We think that the reason is the inherent handicap of AESA to fixed-pivot set methods as discussed before. The distance relationship between query objects and the pivots become more important than inter-pivot distances. This makes it difficult to determine which distances are more important, since query objects are not known at the time of construction.

We believe that Kvp and KvpDD are more appropriate for higher cardinalities because they have linear construction and space complexities, and they restrict the set of pivots that will be used for queries, causing the prioritization to be based on more predictable data. Pivot selection techniques can make sure pivots are far from each other, increasing the likelihood that any query object will find sufficiently close and far pivots from it. Fig. 4 show that KvpDD is able

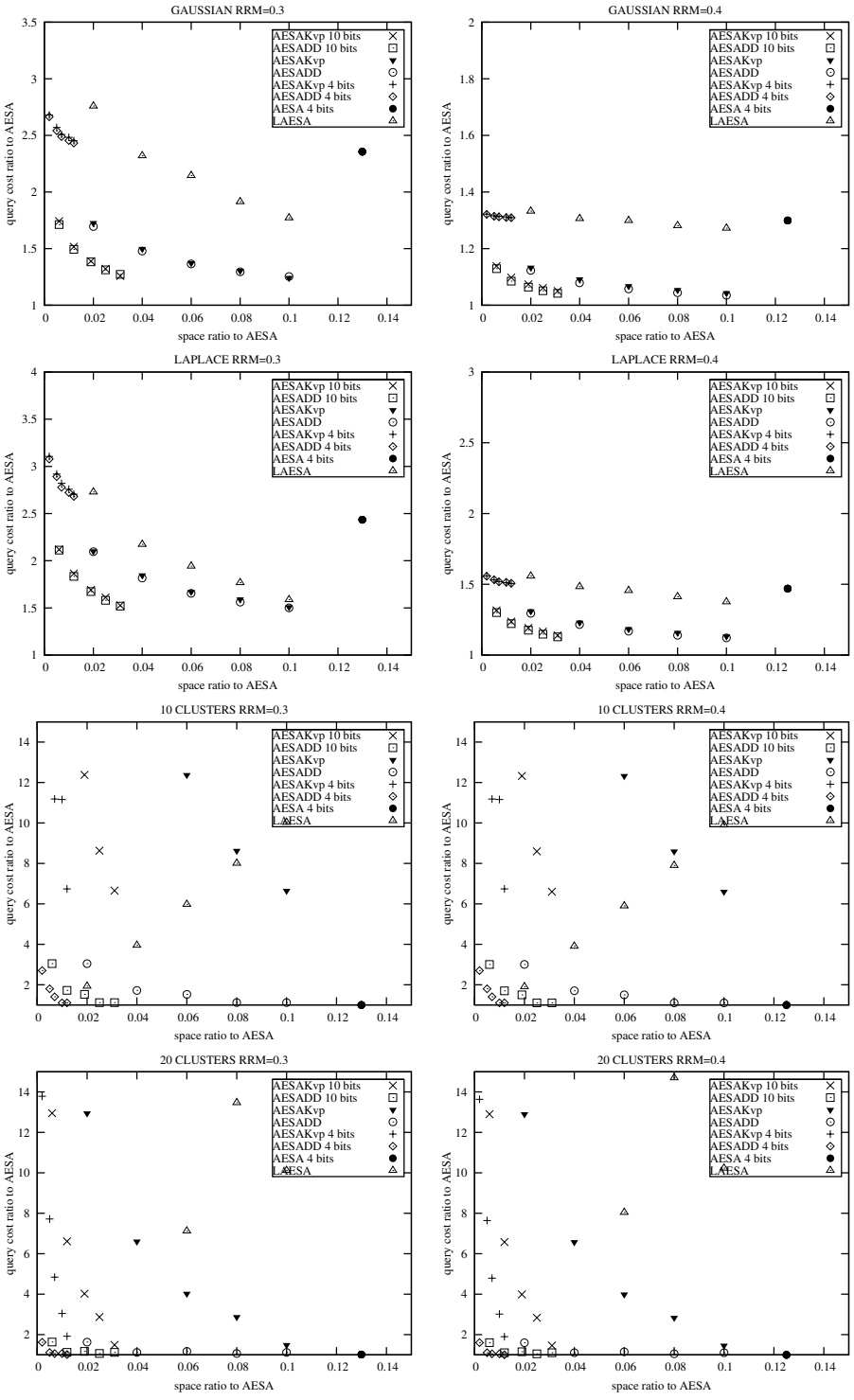

Figure 3. Comparison of AESA variants in various types of vector distributions in 50 dimensions for a database size of 2000

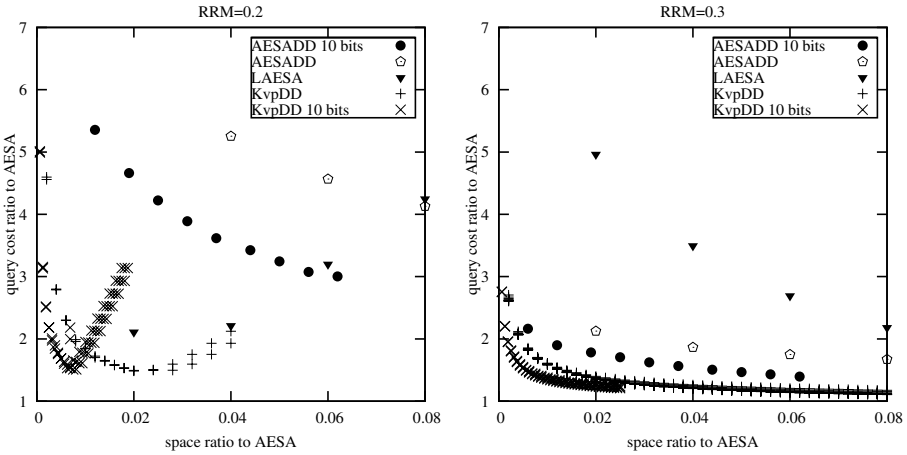

Figure 4. Comparison of DD variants for 10000 uniformly distributed random vectors in 50 dimensions

to provide very competitive trade-off points for the database of 10000 vectors.

We have compared the space usage of KvpDD to some of the other indexing structures in literature. We obtained the implementations for the M-tree, Slim-tree, DF-tree and the DBM-tree from the Arboretum project (http://gbdi.icmc.usp.br/arboretum/). This library uses a word size of 64 bits for distances. For Vp-tree and GNAT we computed the size of the trees by counting references and distances as 8 bytes. For LAESA and Kvp variants, we counted every distance as contributing 10 bytes to the total storage. We use an extra 2 bytes to store the id of the pivot each distance belongs to.

We will follow the same strategy as before to evaluate the space requirements of these structures. We will treat them as offering different space-query performance trade-off points. We can vary the outcome of the tree structures by using different out-degrees for the internal nodes. For disk-based ones, this is accomplished indirectly by varying the page sizes. For LAESA, this can be accomplished by trying different number of pivots. For Kvp and KvpDD, we vary the number of pivots used and the number of distances stored per object independently. This causes multiple performance points in our figures for the same space ratio.

Fig. 5 shows our results. The prioritization schemes offer the best performance. It is also possible to reduce the space usage of KvpDD by more than one third while decreasing the query performance only marginally.

Note that this comparison is not entirely fair. In a disk-based scheme nodes are not fully used, although the minimum occupancy of nodes can be increased at the cost of higher insertion/deletion times. Using a fixed page size introduces some left-over space even at nodes that are completely full. However, by varying the page size parameter, and using especially large page sizes as we did, we can minimize this effect. The most important factor that increases the total space consumption of disk-based methods is that internal nodes need to store objects for navigation. This way, an object can potentially have multiple copies.

4. Conclusions

In this paper we aimed at reducing the space requirements of global pivot-based methods. We argued that pivot-based methods should only store relevant pivot distances. We pointed out the weakness of the Kvp structure, and demonstrated how we can overcome it by computing some simple statistics. Our new structure is very space-efficient, and significantly outperforms the Kvp prioritization scheme for clustered datasets.

We also combined our method with the range coarsening method to increase our space efficiency. Our results show that decreasing the precision of distance values to 10 bits bring only a negligible performance penalty.

We observed that our AESA variants lose their efficiency for higher cardinalities, where incidentally AESA itself becomes infeasible due to its quadratic complexity. We showed that KvpDD is a better option in this case, providing better performance per space compared to tree-based structures.

Our experiments showed that KvpDD is not superior to Kvp in higher cardinalities. Our scheme is based on the assumption that the efficiencies of pivots are independent. However, as the pivots become scarce compared to the size of database, the distance relationships between pivots become important. If two pivots are relatively close, and one

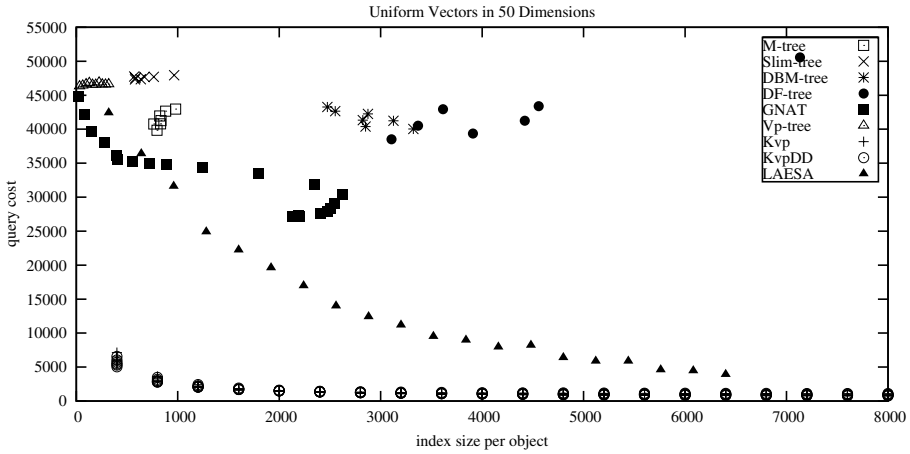

Figure 5. Comparison of space requirements and corresponding query performances for RRM=0.244 in 50 dimensions for 50000 uniform vectors

of them has already been processed, the second one will also offer similar information. We plan to work on this problem by taking the distances between pivots into account when deciding which pivot distances to keep during prioritization.

References

[1] T. Bozkaya and M. Ozsoyoglu. Distance-based indexing for high-dimensional metric spaces. In *SIGMOD '97: Proceedings of the 1997 ACM SIGMOD international conference on Management of data*, pages 357–368, 1997.

[2] S. Brin. Near neighbor search in large metric spaces. In *The VLDB Journal*, pages 574–584, 1995.

[3] B. Bustos, G. Navarro, and E. Chávez. Pivot selection techniques for proximity searching in metric spaces. *Pattern Recognition Letters*, 24(14):2357–2366, 2003.

[4] J. C. Traina, A. J. M. Traina, R. F. S. Filho, and C. Faloutsos. How to improve the pruning ability of dynamic metric access methods. In *CIKM*, pages 219–226, 2002.

[5] J. C. Traina, A. J. M. Traina, B. Seeger, and C. Faloutsos. Slim-trees: High performance metric trees minimizing overlap between nodes. In *Advances in Database Technology - EDBT 2000, 7th International Conference on Extending Database Technology, Konstanz, Germany, March 27-31, 2000, Proceedings*, volume 1777 of *Lecture Notes in Computer Science*, pages 51–65. Springer, 2000.

[6] C. Celik. Priority vantage points structures for similarity queries in metric spaces. In *Proceedings of EurAsia-ICT*, volume 2510 of *Lecture Notes in Computer Science*, pages 256–263. Springer, 2002.

[7] E. Chávez, J. L. Marroquín, and G. Navarro. Fixed queries array: A fast and economical data structure for proximity searching. *Multimedia Tools Appl.*, 14(2):113–135, 2001.

[8] E. Chávez, G. Navarro, R. Baeza-Yates, and J. L. Marroquín. Searching in metric spaces. *ACM Comput. Surv.*, 33(3):273–321, 2001.

[9] P. Ciaccia, M. Patella, and P. Zezula. M-tree: an efficient access method for similarity search in metric spaces. In *Proc. 23rd Conf. on Very Large Databases (VLDB'97)*, pages 426–435, 1997.

[10] K. Figueroa and K. Fredriksson. Simple space-time tradeoffs for aesa. In *WEA*, pages 229–241, 2007.

[11] L. Micó, J. Oncina, and E. Vidal. A new version of the nearest-neighbour approximating and eliminating search algorithm (aesa) with linear preprocessing time and memory requirements. *Pattern Recognition Letters*, 15:9–17, 1994.

[12] G. Navarro, R. Paredes, and E. Chávez. t-spanners for metric space searching. *Data & Knowledge Engineering*, 63(3):818–852, 2007.

[13] J. K. Uhlmann. Satisfying general proximity/similarity queries with metric trees. *Inf. Process. Lett.*, 40(4):175–179, 1991.

[14] E. Vidal. An algorithm for finding nearest neighbors in (approximately) constant average time. *Pattern Recognition Letters*, 4:145, 1986.

[15] E. Vidal. New formulation and improvements of the nearest-neighbour approximating and eliminating search algorithm (aesa). *Pattern Recogn. Lett.*, 15(1):1–7, 1994.

[16] M. R. Vieira, J. C. Traina, F. J. T. Chino, and A. J. M. Traina. Dbm-tree: A dynamic metric access method sensitive to local density data. In S. Lifschitz, editor, *SBBD*, pages 163–177. UnB, 2004.

[17] P. Yianilos. Data structures and algorithms for nearest neighbor search in general metric spaces. In *Proc. 4th ACM-SIAM Symp. on Discrete Algorithms (SODA'93)*, pages 311–321. SIAM Press, 1993.