CIRCULAR DATA-SPACE PARTITIONING FOR SIMILARITY QUERIES AND PARALLEL DISK ALLOCATION

ÖMER EĞECIOĞLU* and HAKAN FERHATOSMANOĞLU
Department of Computer Science
University of California at Santa Barbara
{omer,hakan}@cs.ucsb.edu

Abstract

In a multiple disk environment it is desirable to have techniques for efficient parallel execution of similarity queries. Usually many buckets that may have the query result are needed to be retrieved from secondary storage, which is a costly operation. To achieve efficiency, there are two major factors that need to be considered. These are the number of buckets retrieved by the query, and the degree of parallelism provided by the disk allocation method. In this paper, we develop efficient techniques for parallel similarity searching by optimizing these two factors defined for data-sets that are circular in nature, and similarity queries consisting of query spheres centered at the query point. Our partitioning technique minimizes the expected number of buckets retrieved by a random query among a spectrum of partitioning schemes which have equi-area concentric rings and equi-area central wedges as its two extremes. A simple disk allocation technique for the proposed partitioning method that maximizes the degree of parallelism obtained is also described.

Keywords: Circular partitioning, similarity query, parallel search, multiple disks, disk allocation.

1. Introduction

The volume of multidimensional data that need to be processed has been increasing rapidly. Commercial data warehouses are doubling their size every 9-12 months and satellite data repositories will soon add one to two terabytes of data in a day [1]. If the current trends continue, large organizations will have petabytes of storage managed by thousand of processors [7]. Several applications using this data require efficient support for range and similarity searching. Example of these applications include Geographical Information Systems (GIS) [8], Multimedia Information Systems [11], CAD/CAM [6], medical imaging [10]. General approach is to represent the data objects as multidimensional points and to measure similarity between objects by some notion of distance between the corresponding multidimensional points. Generally, it is assumed that the closer the points, the more similar the data objects. Several index structures have been proposed for retrieval of multidimensional data. Examples of these include kdb-trees, hB-tree, R-tree, R*-tree, SS-tree, TV-tree, X-tree, and the Pyramid Technique [14, 2, 19, 5, 4]. However, traditional retrieval methods based on index structures developed for single disk and single processor environments may be ineffective for the storage and retrieval of multidimensional data in multiprocessor and multiple disk environments. Therefore, it is essential to develop techniques that are optimized for such environments. Note that the buckets that may have the query result are needed to be retrieved from secondary storage as a result of the query. For efficient execution of queries in these environments, there are two major factors that need to be considered:

1. the number of buckets retrieved by the query,

2. the degree of parallelism provided.

In this paper, we focus on similarity queries. One of the typical query types is the ϵ-similarity which is specified by a query point and a radius which defines the acceptable region of similarity. Another typical query is the k-nearest neighbor query in which k most similar objects to the query object need to be reported. Both of these query types need to retrieve a spherical region as a result whereas rectangular range queries retrieve a rectangular region.

For range and ϵ-similarity queries the buckets, i.e. the subdivisions of the data-set, that intersect the query region

*Supported in part by NSF Grant no. IIS-9817432.

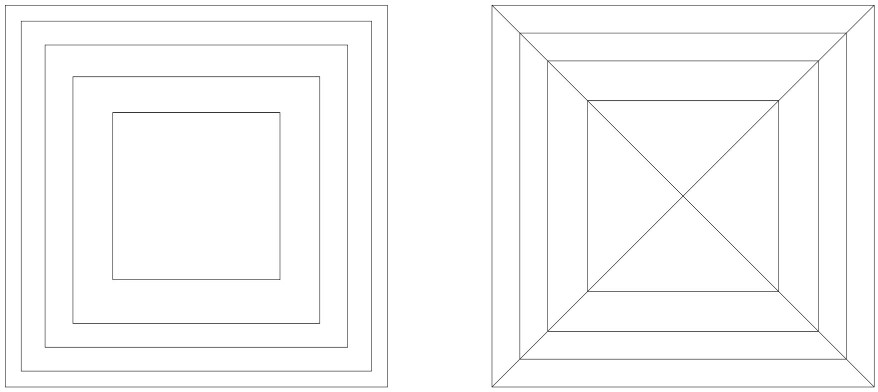

Figure 1. Concentric hyper-cubes and hyper-pyramids

are needed to be retrieved. It is evident that the number of buckets retrieved by a query is important in the performance of a query and this number directly depends on the underlying partitioning strategy [13, 4]. If we run the same query in two differently partitioned data sets, we may end up retrieving different number of buckets depending on the way the data-space is initially organized. Hence, it is desirable to develop efficient partitioning techniques to minimize the expected number of buckets retrieved as result of similarity queries. The first parameter is the result of this fact. These buckets are retrieved from parallel disks in a multi-disk environment. The degree of parallelism effects the performance of the query result. The core problem in parallel search is to distribute the buckets among several I/O devices such that the data retrieved by any query is evenly spread across all the I/O devices. For example, if all the buckets retrieved as a result of a given query are allocated to different devices, the degree of parallelism, and therefore the query execution time is minimized. However, if they are allocated to the same device, the buckets are retrieved from one single disk sequentially. The second parameter, degree of parallelism, must be maximized by developing appropriate allocation techniques for the underlying partitioning.

2. Problem statement

In this paper, we develop efficient techniques for parallel similarity searching in spherical data-sets by optimizing the two important factors defined above for parallel searching. We develop and analyze appropriate partitioning techniques which minimize the number of buckets retrieved by any query. Then, we develop disk (or I/O device) allocation techniques that maximize the degree of parallelism.

Concentric partitioning is shown to be useful for parallel searching. The idea of using concentric partitioning for parallel searching is proposed in [13]. Two different partitioning strategies, concentric hyper-cubes and hyper-pyramids shown in Figure 1 are discussed in the context of parallel execution of rectangular range queries. These techniques are useful for declustering optimized for rectangular range searching. In this paper, we focus on ϵ-similarity queries. We first develop theoretical analysis to find the optimal way of circular data-space partitioning which optimizes the first parameter discussed in the previous section. For hyper-rectangular range queries, the hyper-pyramid partitioning is obtained by dividing concentric hyper-cubes into $2d$ more divisions from the center through the edges, where d is the number of dimensionality. For similarity searching, we will consider spherical queries, and spherical partitions. The overall region of the data-space is also taken to be a sphere. One extreme possibility is to partition the data-space by concentric spheres (rightmost partition in Figure 2). Other extreme way is to partition the data-space into wedges without using any concentric rings (leftmost partition in Figure 2). These two ways are similar to the concentric hyper-cubes and hyper-pyramids, respectively. Here, we will explore the possibilities of having partitioning strategies in between of these extremes, as illustrated in Figure 2. Is it feasible and possible to build a partitioning technique optimized for the first parameter for similarity queries which also achieves efficient parallelism? After stating the appropriate partitioning technique, we develop an allocation method for the proposed partitioning to achieve efficient degrees of parallelism.

3. Theoretical considerations

For the analysis of efficient partitioning techniques for similarity searching, we first start with the observation that there is a relation between the the expected number of partitions (we also refer to these as buckets, subdivisions, or regions) a spherical query intersects with the total boundary of the partitions. We concentrate on the two dimensional case and assume that the expected number of regions a small disc

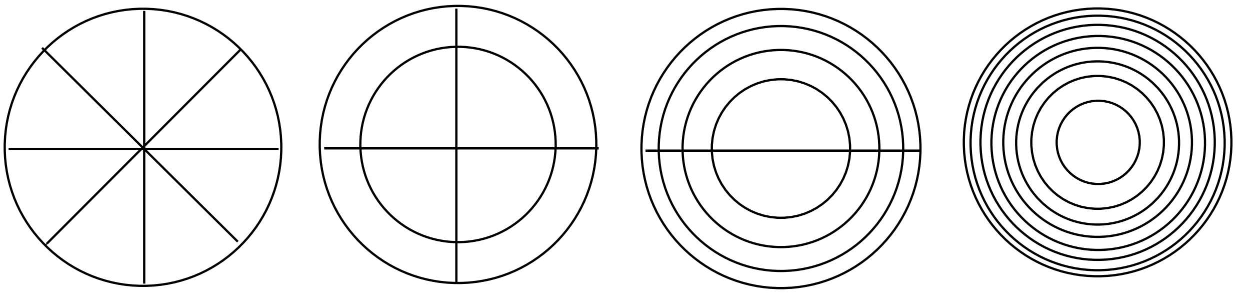

Figure 2. Equi-area subdivision of a circular data-space into $n = 2^k$ regions consisting of 2^r rings and 2^{k-r} central wedges: $k = 3$, $r = 0, 1, 2, 3$.

of radius ρ intersects is minimized when the total boundary of the regions forming the partition is minimized. This is not exactly right if ρ is not small, since the number of regions meeting at a point is important in the calculation of the expected value, whereas the length of the boundary does not directly take this into account (i.e. it treats the number of regions intersected as either 1 (one side of a boundary line) or 2 (intersecting a boundary line)). However for small ρ, we argue that this assumption is valid as follows: Consider a strip of width 2ρ around each boundary line, with the boundary line running at the center as shown in Figure 3. Any query circle of radius ρ centered outside the region

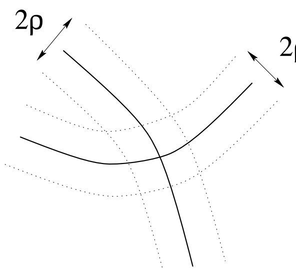

Figure 3. Crossing partition boundaries.

S in the unit circle formed by these strips is contained in a single region of the subdivision. If the total area of S is A, then the expected number E of regions intersected by a random query circle of radius ρ satisfies

$$\frac{1}{\pi}(\pi - A) + 2\frac{A}{\pi} \leq E \leq \frac{1}{\pi}(\pi - A) + n\frac{A}{\pi}.$$

This is because each query circle with center in S intersects at least 2 and at most n regions of the subdivision. Thus

$$1 + \frac{A}{\pi} \leq E \leq 1 + (n-1)\frac{A}{\pi}, \tag{1}$$

and for any fixed n, E approaches its minimum possible value of 1 as A approaches 0. Therefore for small enough ρ, the expected value E is minimized for a subdivision into equal parts with minimum boundary. This is a type of an isoperimetric problem. Note that in (1), it is possible to use planarity and reduce the quantity $n - 1$ for boundaries that are not pathological. This is because by Euler's formula the average degree of a vertex in a planar graph is no larger than 6.

Now we assume that the query radius ρ is small, and calculate the value of r which gives the optimal subdivision into n regions. We assume that $n = 2^k$ for some integer $k \geq 0$. We also assume that the query disc is completely contained in the unit disc. In other words, its center lies in the disc with origin as center and of radius $1 - \rho$. Thus the boundary of the unit circle itself need not be taken into account in these calculations.

The types of subdivisions of the unit disc into $n = 2^k$ regions we choose to consider are parameterized by r, and corresponding to r, the unit disk is first divided into 2^r equal-area concentric rings. Then each of these rings is further divided up by 2^{k-r} central wedges of equal angles for a total of 2^k equal area regions. Which value of $r = 0, 1, \ldots, k$ minimizes the total boundary?

The radius x_1 of the innermost disc of the division into 2^r rings is found from $\pi x_1^2 = \pi/2^r$ as $x_1 = 1/\sqrt{2^r}$. Similarly, the radius of the i-th innermost disc is $x_i = \sqrt{i}/\sqrt{2^r}$ with perimeter $2\pi\sqrt{i}/\sqrt{2^r}$. The sum of the perimeters of the $2^r - 1$ circles is then

$$\frac{2\pi}{\sqrt{2^r}}(1 + \sqrt{2} + \sqrt{3} + \cdots + \sqrt{2^r - 1}).$$

To this sum, we add the lengths of the 2^{k-r} radii that form the boundary of the wedges, to obtain an expression for the total boundary in terms of r as

$$\frac{n}{2^r} + \frac{2\pi}{\sqrt{2^r}}(1 + \sqrt{2} + \sqrt{3} + \cdots + \sqrt{2^r - 1}). \tag{2}$$

To get an idea about the magnitude of r that minimizes this expression, we approximate

$$1+\sqrt{2}+\sqrt{3}+\cdots+\sqrt{x-1} \approx \int_0^x \sqrt{t}\,dt = \tfrac{2}{3}x^{3/2}$$

and minimize the real-valued function of x given by

$$\frac{n}{x} + \frac{4\pi}{3\sqrt{x}}\,x^{3/2}$$

on $1 \le x \le n$. By calculus, we find that the minimum is achieved at

$$x = \frac{1}{2}\sqrt{\frac{3}{\pi}}\sqrt{n} \approx 0.489\sqrt{n}.$$

This means that the optimal exponent r for $n = 2^k$ is roughly

$$r \approx \tfrac{1}{2}k - 1. \qquad (3)$$

For small values of k and $n = 2^k$, the optimal values of r that minimizes the expression in (2) can be calculated numerically. The corresponding value of 2^r is the number of rings, and $n/2^r$ is the number of wedges that the unit disc needs to be divided into to minimize the boundary of the n equal-area regions. The values of the optimal exponent r calculated by brute force from (2) for $k \le 19$ are given in the following table.

k	1	2	3	4	5	7	8	9	10
r	0	0	1	1	2	2	3	3	4

k	11	12	13	14	15	16	17	18	19
r	4	5	5	6	6	7	7	8	8

These values are in perfect agreement with the formula for r given in (3). In particular when n is of the form $n = 4^k$, then the optimal boundary subdivision has 2^{k-1} rings and 2^{k+1} wedges.

4. Implementation

When we subdivide the unit disk in the plane into 2^k regions by first dividing into 2^r equi-area rings, and then dividing each ring into 2^{k-r} by means of equi-area central wedges, each region in the subdivision is determined by 4 parameters. A pair of angles θ_1, θ_2 determines the wedge that the region is in, and a pair of radii r_1, r_2 determines which ring the region is in. The angles satisfy $0 \le \theta_1 < \theta_2 \le 2\pi$. The boundary case $\theta_1 = 0$, and $\theta_2 = 2\pi$ is interpreted as the subdivision in which there are no wedges (i.e. $r = k$ and the regions consist only of rings). The two radii satisfy $0 \le r_1 < r_1 \le 1$. The extreme cases $r_1 = 0$ and $r_2 = 1$ correspond to the subdivision in which there are no rings (i.e. $r = 0$ and the regions consist only of wedges).

The regions are labeled from 0 to $2^k - 1$ as follows. First of all, label the 2^r rings from 0 to $2^r - 1$ by increasing radius as we go out from the origin to the boundary of the unit circle. In each ring, label the 2^{k-r} pieces determined by the wedges from 0 to $2^{k-r} - 1$ counterclockwise, starting with the wedge that is incident to the horizontal axis in the first quadrant. An integer m with $0 \le m < 2^k$ can be written uniquely in the form $m = q2^r + s$ with $0 \le q < 2^{k-r}$, and $0 \le s < 2^r$. The pair (q, s) uniquely corresponds to the region in the q-th wedge of the s-th ring of the subdivision under this numbering scheme. For example the region labeled 7 in Figure 4 is encoded as the pair $(3, 1)$, since $7 = 3 \times 2 + 1$. Aside from the extreme cases of $r = 0$ and $r = k$, the boundary of the m-th region is described analytically by the radii

$$r_1 = \frac{\sqrt{s}}{\sqrt{2^r}}, \quad r_2 = \frac{\sqrt{s+1}}{\sqrt{2^r}},$$

and the two angles

$$\theta_1 = \frac{2\pi q}{2^{k-r}}, \quad \theta_2 = \frac{2\pi(q+1)}{2^{k-r}}.$$

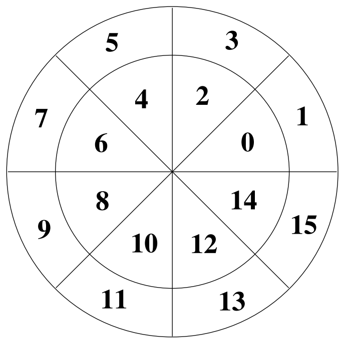

Figure 4. Labeling of the subdivisions: $n = 2^4$ **and** $r = 1$**.**

5. Experimental results

We have conducted experiments to calculate the expected number of regions intersected by a randomly selected query circle of radius ρ in the unit circle for varying values of ρ. The experiments for the calculation of the expected values were conducted with three parameters: $n = 2^k$, r, and the radius ρ of the query circle. The data-space is divided into 2^k regions by first dividing into 2^r equi-area rings, and then dividing each ring into 2^{k-r} by means of equi-area central wedges, $r = 0, 1, \ldots, k$.

In the first set of experiments, we chose the number of partitions to be $n = 2^{12}$, i.e. $k = 12$. When $r = 0$, then the partitioning is just the set of wedges without any concentric rings, i.e., the leftmost partition in Figure 2. When $r = 12$, the partitions are formed by circles only, creating 2^{12} concentric rings as partitions, analogous to the rightmost partition in Figure 2. In the experiments the query center is chosen randomly in the data-space from the uniform distribution in such a way that the query region lies entirely within the data-space.

We started with the value of the query radius $\rho_0 = \frac{1}{\sqrt{n}}$, which gives the query circle the same areas as the area of each region. Figure 5 illustrates the results of the experiments run with this initial value of ρ. The horizontal axis gives the values of r ranging from 0 to 12, and the vertical axis is the average number of partitions intersected by a random query of query radius ρ. This quantity is proportional to the cost of retrieval from secondary storage, since the cost of a query depends on the number of buckets retrieved as a result of the query, and this is exactly the number of partitions which intersect the query circle. When $r = 4$ and $r = 5$ the number of intersected partitions is found to be minimized with expected number $E = 6.55$ and $E = 6.65$, respectively. This result is in agreement with the theoretical analysis. When $r = 12$, the average number of partitions intersected by the queries is $E = 118.55$, about 18 times greater than the average number of regions intersected when $r = 5$.

We vary the radius of the query. The initial value of $\rho_0 = \frac{1}{\sqrt{n}}$ was reduced by a factor of $\frac{1}{2}$ in in each subsequent set of experiments. The corresponding radii are $\rho_1 = \frac{1}{2\sqrt{n}}$, $\rho_2 = \frac{1}{2^2\sqrt{n}}$ etc., until the smallest radius value $\rho_4 = \frac{1}{2^4\sqrt{n}}$. For each radius value ρ, we generate random queries and find the number of intersections of the query circle of radius ρ centered at the query point generated with the regions in the partition. In Figure 6 the horizontal axis is ρ_i, where $0 \leq i \leq 4$, and the vertical axis is the average number of partitions retrieved by the queries in the case of (1) optimal way of partitioning, (2) partitioning based on our approximation, (3) fully-wedged partitioning (leftmost partition in Figure 2), and (4) fully-concentric partitioning (rightmost partition in Figure 2). Partitioning technique based on theoretical analysis gives optimal cost, i.e optimal number of intersected partitions, for most of the queries. Our theoretical analysis is in perfect agreement with the experiments. The number of partitions retrieved by the queries in concentric and fully-wedged partitioning is much more than the hybrid approach with the appropriate parameters developed in the previous sections.

6. Disk allocation methods

In order to obtain a good degree of parallelism careful distribution of buckets to multiple disks is necessary. There have been several disk allocation techniques for regular grid partitioning. The partitioning technique described here is different from the current techniques used for declustering, therefore we can not use them directly. However, it is possible to apply the intuitions behind the current approaches here. As in the general case, we concentrate on the neighboring buckets and distribute them to different disks. The direct neighbors are defined as the partitions that have a common boundary, e.g. in Figure 4 the direct neighbors of the partition 2 are 0, 3, and 4. The indirect neighbors are defined as the partitions that share a point, e.g. the indirect neighbors of partition 3 are partitions 0 and 4.

Our data-space is partitioned into 2^k equi-area regions determined by 2^r concentric rings and 2^{k-r} central wedges where $r = \lfloor \frac{1}{2}k \rfloor - 1$. As mentioned before, each partition corresponds to an ordered pair of integers (q, s), where s is the rank of the ring ($0 \leq q < 2^r$), and q is the rank of the wedge inside that ring ($0 \leq s < 2^{k-r}$). For example, in Figure 4 $(0, 0)$ corresponds to the partition labeled 0, the pair $(2, 1)$ corresponds to the partition labeled 5, and $(5, 0)$ corresponds to the partition 10. In general, the pair (q, s) represents the partition labeled $q2^r + s$. The allocation technique we propose is as follows. Given the number M of available disks, we use a generic allocation technique parametrized by a skip value H. The partition $(0, 0)$ is assigned to device 0. Next, the partitions along this ring (with rank $s = 0$) is assigned to consecutive devices, i.e. partition $(q, 0)$ is assigned to device $q \bmod M$. Each partition labeled $(q, 1)$ in ring 1 is assigned to device $(H+q) \bmod M$, which is H devices away from the device on which the partition from the same wedge level in ring 0 was assigned. In general, a partition (q, s) on ring s is assigned to device $(Hq + s) \bmod M$. This assignment is an extension of the Cyclic Allocation Technique applied to our partitioning. Cyclic allocation was originally proposed for regular grid partitioning and its performance depends on the H value that is used. The techniques discussed in [17] to determine appropriate H values can also be used here.

In addition to cyclic allocation, there have been a number of other disk allocation techniques proposed for regular grid partitioning, especially in the context of relational data and partial match and range queries. Disk Modulo (DM) [9], Fieldwise Exclusive (FX) [16], Hilbert (HCAM) [12], Near Optimal Declustering (NoD) [3], General Multidimensional Data Allocation (GMDA) [15], Cyclic Allocation Schemes [17, 18] are the well-known techniques. All of these can be applied to our partitioning scheme.

There are two different metrics to evaluate the degree of parallelism for similarity searching. In [3], the goal of allocation technique is defined to ensure that any two buckets that are direct or indirect neighbors of each other are allocated to different disks. Such an allocation is defined to be *near-optimal*. The second metric is defined in [18] as

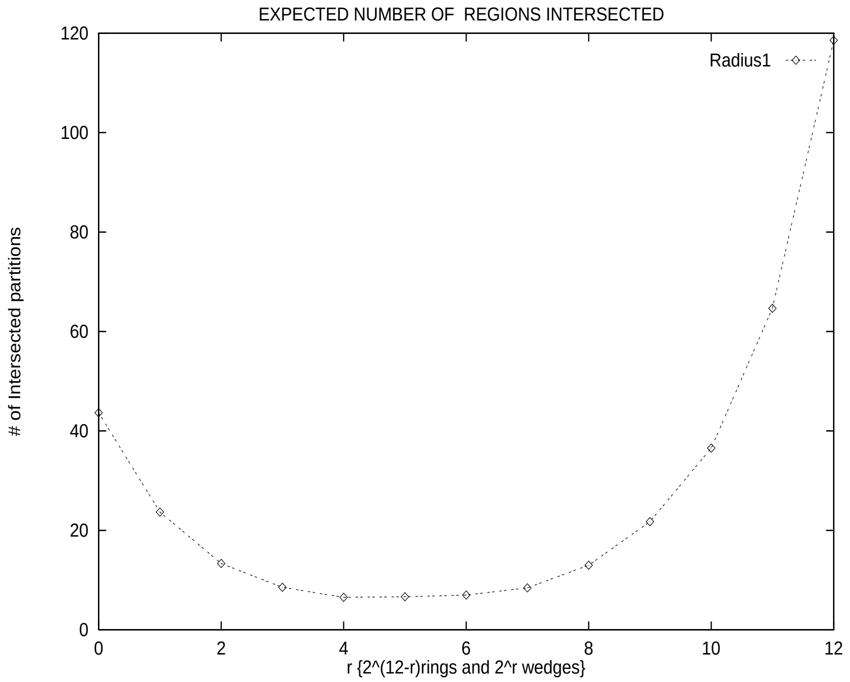

Figure 5. Effect of the number of wedges versus rings on the expected number of of intersected partitions. Query radius is $\rho_0 = 1/\sqrt{n}$.

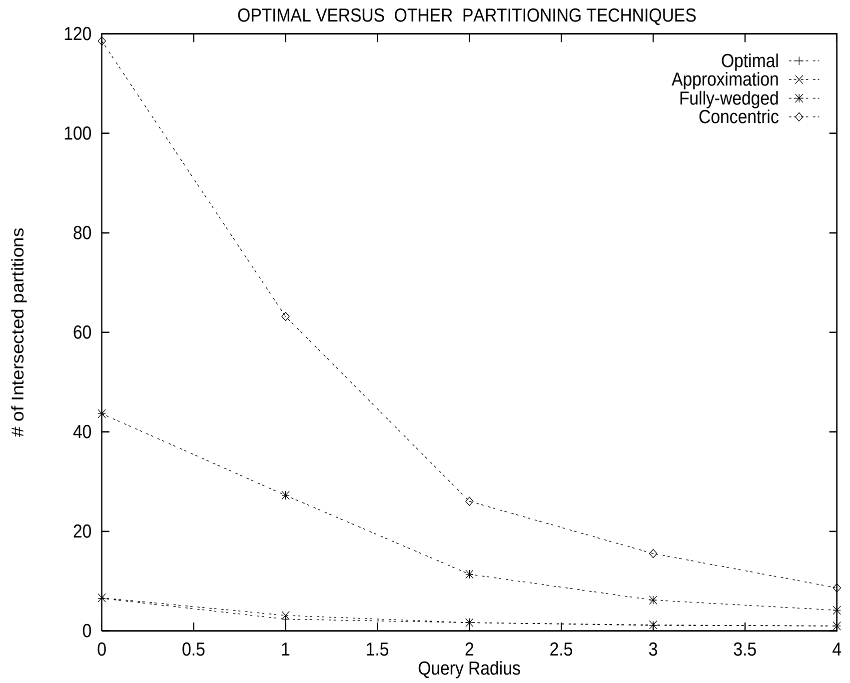

Figure 6. The performance of different partitioning techniques with varying query radius. The point i on the horizontal axis denotes query radius $\rho_i = \frac{1}{2^i\sqrt{n}}$.

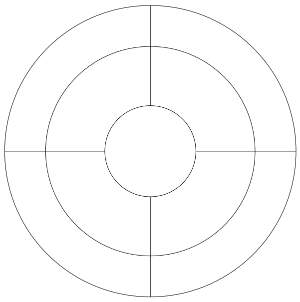

Figure 7. Additional partition in the center

follows. The maximum degree of parallelism is achieved when buckets that are retrieved together are spread among all available disks as uniformly as possible. The cyclic technique is shown to be efficient in terms of these two different metrics. In fact, it has been shown that by using cyclic allocation it is possible to find appropriate skip values such that no two direct or indirect neighbors are allocated to the same disk if the data-space is $d-$dimensional space and $2d$ disks are available. Except for the border and inside most partitions, e.g. $0, 2, 4, \ldots, 14$, the direct and indirect neighboring bucket pairs remain same in regular grid partitioning and in the partitioning proposed here. Since cyclic technique guarantees the distribution of the direct and indirect neighbors to different disks in regular grid partitioning, our allocation technique on the new partitioning scheme also guarantees this. Therefore, disk allocation technique described in this section is also near-optimal. Similarly, we expect the same good performance of cyclic allocation in this context. To avoid the neighboring problem for the innermost partitions, we can just add one more partition in the center of the data-space. This additional partition will add two advantages. First, we eliminate the case that a single point, i.e. the center, is contained in all 2^{n-r} central wedges. This eliminates the retrieval of all 2^{n-r} innermost partitions for a a query point involving the center. Second, the innermost partitions other than the adjacent ones are no longer direct neighbors. Figure 7 illustrates this possible extension.

7. Conclusions

There are two major factors that determine the efficiency of ϵ-similarity queries in a multiple disk environment: the number of buckets retrieved by the query circle, and the the degree of parallelism provided by the disk allocation method. The idea is to minimize the expected number of buckets retrieved by a random query and at the same time devise an efficient disk allocation scheme for the resulting partitioning that maximizes the degree of parallelism obtained.

We have described techniques for parallel similarity searching in circular data-sets in the plane meeting these two conditions: the number of buckets retrieved by the query is minimized by an appropriate selection of partitioning, and the degree of parallelism is maximized by applying disk (or I/O device) allocation techniques suitable for the partitioning selected. The partitions considered form a spectrum in which the equi-area concentric rings and equi-area central wedges form extreme cases. We showed that the optimal partitioning into n buckets uses a mixture of about $\sqrt{n}$ of each of these types of regions, and conducted experiments with random queries without boundary effects. The findings obtained by varying the query-radii for different partitions and query points generated from the uniform distribution and calculating the average number of buckets intersected support the theoretical findings. For the general case, the construction of spherical shells by means of concentric spheres in high dimensions is straightforward, whereas the the analogue of the wedge-shaped regions requires more care. A wedge is determined by a pair of angles in the plane, but the analogous region in three dimensions requires two pairs of angles, and in d dimensions, $d-1$ pairs of angles. Therefore for the d- dimensional spherical/wedge partitions, each bucket will have a description consisting of d pairs of numbers.

References

[1] A. Acharya, M. Uysal, and J. Saltz. Active disks: Programming model, algorithms and evaluation. In *ASPLOS-VIII*, pages 81–91, Sept. 1998.

[2] N. Beckmann, H. Kriegel, R. Schneider, and B. Seeger. The R* tree: An efficient and robust access method for points and rectangles. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 322–331, May 23-25 1990.

[3] S. Berchtold, C. Bohm, B. Braunmuller, D. A. Keim, and H.-P. Kriegel. Fast parallel similarity search in multimedia databases. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 1–12, Arizona, U.S.A., 1997.

[4] S. Berchtold, C. Bohm, and H.-P. Kriegel. The Pyramid-Technique: Towards breaking the curse of dimensionality. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 142–153, Seattle, Washington, USA, June 1998.

[5] S. Berchtold, D. A. Keim, and H. P. Kreigel. The X-tree: An index structure for high-dimensional data. In *22nd. Conference on Very Large Databases*, pages 28–39, Bombay, India, 1996.

[6] K. H.-P. Berchtold S. S3: Similarity search in cad database systems. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 564–567, Tuscon, Arizona, 1997.

[7] P. Bernstein, M. Brodie, S. Ceri, D. DeWitt, M. Franklin, H. Garcia-Molina, J. Gray, J. Held, J. Hellerstein, H. Jagadish, M. Lesk, D. Maier, J. Naughton, H. Pirahesh, M. Stonebraker, and J. Ullman. The Asilomar report on database research, December 1998.

[8] X. Cheng, R. Dolin, M. Neary, S. Prabhakar, K. Ravikanth, D. Wu, D. Agrawal, A. El Abbadi, M. Freeston, A. Singh, T. Smith, and J. Su. Scalable access within the context of digital libraries. In *IEEE Proceedings of the International Conference on Advances in Digital Libraries, ADL*, pages 70–81, Washington, D.C., 1997.

[9] H. C. Du and J. S. Sobolewski. Disk allocation for cartesian product files on multiple-disk systems. *ACM Transactions of Database Systems*, 7(1):82–101, March 1982.

[10] F. Korn, N. Sidiropoulos, C. Faloutsos, E. Siegel, and Z. Protopapas. Fast nearest neighbor search in medical image databases. In *Proceedings of the Int. Conf. on Very Large Data Bases*, pages 215–226, Mumbai, India, 1996.

[11] C. Faloutsos, R. Barber, M. Flickner, J. Hafner, W. Niblack, D. Petkovic, and W. Equitz. Efficient and effective querying by image content. *Journal of Intelligent Information Systems*, 3:231–262, 1994.

[12] C. Faloutsos and P. Bhagwat. Declustering using fractals. In *Proceedings of the 2nd International Conference on Parallel and Distributed Information Systems*, pages 18 – 25, San Diego, CA, Jan 1993.

[13] H. Ferhatosmanoglu, D. Agrawal, and A. E. Abbadi. Concentric hyperspaces and disk allocation for fast parallel range searching. In *Proc. Int. Conf. Data Engineering*, Sydney, Australia, Mar. 1999.

[14] V. Gaede and O. Gunther. Multidimensional access methods. *ACM Computing Surveys*, 30:170–231, 1998.

[15] K. A. Hua and H. C. Young. A general multidimensional data allocation method for multicomputer database systems. In *Database and Expert System Applications*, pages 401–409, Toulouse, France, Sept. 1997.

[16] M. H. Kim and S. Pramanik. Optimal file distribution for partial match retrieval. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 173–182, Chicago, 1988.

[17] S. Prabhakar, K. Abdel-Ghaffar, D. Agrawal, and A. El Abbadi. Cyclic allocation of two-dimensional data. In *International Conference on Data Engineering*, pages 94–101, Orlando, Florida, Feb 1998.

[18] S. Prabhakar, D. Agrawal, and A. El Abbadi. Efficient disk allocation for fast similarity searching. In *10th International Symposium on Parallel Algorithms and Architectures, SPAA`98*, Puerto Vallarta, Mexico, June 1998.

[19] D. White and R. Jain. Similarity indexing with the SS-tree. In *Proc. Int. Conf. Data Engineering*, pages 516–523, 1996.