On Query Processing and Optimality Using Spectral Locality-Preserving Mappings*

Mohamed F. Mokbel Walid G. Aref

Department of Computer Sciences, Purdue University
{mokbel,aref}@cs.purdue.edu

Abstract. A locality-preserving mapping (LPM) from the multi-dimensional space into the one-dimensional space is beneficial for many applications (e.g., range queries, nearest-neighbor queries, clustering, and declustering) when multi-dimensional data is placed into one-dimensional storage (e.g., the disk). The idea behind a locality-preserving mapping is to map points that are nearby in the multi-dimensional space into points that are nearby in the one-dimensional space. For the past two decades, fractals (e.g., the Hilbert and Peano space-filling curves) have been considered the natural method for providing a locality-preserving mapping to support efficient answer for range queries and similarity search queries. In this paper, we go beyond the idea of fractals. Instead, we investigate a locality-preserving mapping algorithm (The Spectral LPM) that uses the spectrum of the multi-dimensional space. This paper provably demonstrates how Spectral LPM provides a globally optimal mapping from the multi-dimensional space to the one-dimensional space, and hence outperforms fractals. As an application, in the context of range queries and nearest-neighbor queries, empirical results of the performance of Spectral LPM validate our analysis in comparison with Peano, Hilbert, and Gray fractal mappings.

1 Introduction

An important consideration for multi-dimensional databases is how to place the multi-dimensional data into a one-dimensional storage media (e.g., the disk) such that the spatial properties of the multi-dimensional data are preserved. In general, there is no total ordering that fully preserves spatial locality. A mapping function f that maps the multi-dimensional space into the one-dimensional space provides a total ordering for the multi-dimensional data. A desirable property for the mapping function f is *locality-preservation*. Mapping data from the multi-dimensional space into the one-dimensional space is considered *locality-preserving* if the points that are nearby in the multi-dimensional space are nearby in the one-dimensional space.

Locality-preserving mappings from the multi-dimensional space into the one-dimensional space are used in many applications, for example:

* This work was supported in part by the National Science Foundation under Grants IIS-0093116, EIA-9972883, IIS-0209120, and by Purdue Research Foundation.

- Range query processing [11, 14, 26, 37]: A locality-preserving mapping enhances the performance of multi-dimensional range queries. In a range query, it is preferable that the qualifying records be located in consecutive blocks rather than being randomly scattered in disk. A good locality-preserving mapping maps the records that lie in the query window in the multi-dimensional space into a consecutive set of blocks in disk.
- Nearest-neighbor finding and similarity search [14, 33, 44]: Multi-dimensional data is stored in disk using a locality-preserving mapping such that the nearest-neighbor for any point P can be retrieved by performing a sequential scan in the forward and backward directions from P. The quality of the locality-preserving mapping algorithms is determined by: (1) the amount of sequential data that needs to be accessed to find the nearest-neighbor and (2) the accuracy of the result.
- Spatial join of multi-dimensional data [38]: Multi-dimensional data is mapped into a one-dimensional domain using a locality-preserving mapping. The transformed data is stored in a one-dimensional data structure, e.g., the B^+-Tree [7], and a one-dimensional spatial join algorithm is applied.
- Spatial access methods [43]: Spatial objects located in disk storage are ordered according to the one-dimensional value of their central point, which is obtained from a locality-preserving mapping of the multi-dimensional space. This mapping minimizes the number of times a given page is retrieved from disk.
- R-Tree Packing [28]: The multi-dimensional central points of a set of rectangles are mapped into a one-dimensional domain using a locality-preserving mapping. Then, the rectangles are packed into the R-Tree [20] based on the one-dimensional value of their central points.
- Other uses of locality-preserving mappings include GIS Applications [5], declustering [12], multi-dimensional indexing [32], multimedia databases [35], disk scheduling [2], image processing [47], the traveling salesman problem [3], and bandwidth reduction for sampling signals [4].

These applications use space-filling curves (SFCs) [42], and fractals [34] to provide locality-preserving mappings. Examples of these curves are the Hilbert SFC [22], the Peano SFC [39], and the Gray SFC [10]. The Hilbert SFC is used for locality-preserving mapping in [13, 14, 26, 28, 32, 33, 43] while the Peano SFC is used for locality-preserving mapping in [5, 38, 47]. The Gray SFC is used for locality-preserving mapping in [10, 11].

In this paper, we go beyond the idea of using fractals as a means of a locality-preserving mapping. Instead, we develop a *Spectral Locality-Preserving Mapping* algorithm (Spectral LPM, for short) [36], that makes use of the spectrum of the multi-dimensional space. Although we focus on the effect of Spectral LPM in enhancing the performance of range queries and nearest-neighbor queries, we believe that Spectral LPM can efficiently replace any of the fractal locality-preserving mappings in the applications mentioned above. The contributions of this paper can be summarized as follows:

1. We argue against the use of fractals as a basis for locality-preserving mapping algorithms and give some examples and experimental evidence to show why the fractal-based algorithms produce a poor mapping (Section 2).
2. We introduce the Spectral LPM algorithm, an optimal locality-preserving mapping algorithm that depends on the spectral properties of the multi-dimensional points. As in the case of fractals, the Spectral LPM algorithm can be generalized easily to any multi-dimensional space (Section 3).
3. We define the notion of global optimality in locality-preserving mappings with respect to all multi-dimensional points, and prove that the Spectral LPM achieves this optimality (Section 4) while fractals do not. Also, we show that there are many cases that are infeasible to be mapped using fractals, while the same cases can be easily mapped optimally using Spectral LPM (Section 5).
4. As an application, in the context of range queries and nearest-neighbor queries, we provide empirical results of the performance of Spectral LPM using real data sets. The performance results validate our analysis in comparison with several fractal mappings. We demonstrate that Spectral LPM is superior to the fractal-based algorithms that have long been used for locality-preserving mappings (Section 6).

The rest of this paper is organized as follows: Section 2 provides the motivation for the spectral-based algorithm by showing the drawbacks of the fractal-based algorithms. Algorithm Spectral LPM is introduced in Section 3. Section 4 gives the proof of optimality of Spectral LPM. Section 5 demonstrates how Spectral LPM may incorporate additional requirements to the locality-preserving mapping. Experimental results comparing the performance of the spectral- and fractal-based mapping algorithms are presented in Section 6. Finally, Section 7 concludes the paper.

2 Locality-Preserving Mappings: The "Good", The "Bad", and The "Optimal"

In this section, we start by describing what properties an optimal locality-preserving mapping algorithm should have, and then discuss whether or not such a mapping is feasible. Next, we discuss the fractal locality-preserving mapping algorithms based on the Peano, Gray and Hilbert space-filling curves. We give examples that show why these algorithms produce a poor mapping. Finally, we discuss the idea of a spectral locality-preserving mapping that avoids the drawbacks of the fractal mapping.

2.1 The "Good" Mapping

An optimal locality-preserving mapping algorithm maps the multi-dimensional space into the one-dimensional space such that the distance between each pair of points in the multi-dimensional space is preserved in the one-dimensional space.

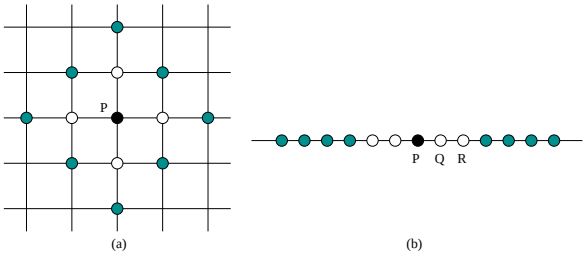


Fig. 1. The Optimal Locality-Preserving Mapping with respect to P.

However, such a mapping is not feasible. For example, for any point P in the D-dimensional space, there are $2D$ neighboring points with Manhattan distance $M = 1$. Mapping P and its neighbors into the one-dimensional space allows only two neighbors to have $M = 1$. Thus, the distance between $2(D-1)$ of the points cannot be preserved. The best we can do in this case is to divide the $2D$ neighbor points into two equal groups with D points in each group, and map the first group to the right of point P in the one-dimensional space, and the other D points to the left of P. The same argument is valid for points with Manhattan distance $M > 1$. The idea of such a mapping is that it guarantees that the points with Manhattan distance k from P in the multi-dimensional space will be nearer to P in the one dimensional space than the points with Manhattan distance $k + 1$.

Figure 1a gives an example of Point P in the two-dimensional space, where it has four neighbors with $M = 1$ (the white points) and eight neighbors with $M = 2$ (the gray points). Mapping P and its white neighbors into the one-dimensional space as in Figure 1b, results in only two points with $M = 1$ and another two points with $M = 2$. Mapping the eight gray neighbors results in placing four of them to the right of P and the other four to the left of P. Such a mapping is considered an optimal locality-preserving mapping with respect to P, since all the points that have $M = 1$ (the white points) in the two-dimensional space are nearer to P in the one-dimensional space than any of the points that have $M = 2$ (the gray points) in the two-dimensional space.

Although this locality-preserving mapping algorithm seems to be simple and optimal with respect to P, the mapping does not guarantee its optimality with respect to any other point. For example, consider the two white points Q, R that have $|P-Q| = 1$, $|P-R| = 2$ in the one-dimensional space as in Figure 1b. With respect to R, in the two-dimensional space, $|R-Q| = 2$ and $|R-P| = 1$. However, in the one-dimensional space, the situation is reversed where $|R - Q| = 1$ and $|R - P| = 2$. Thus, locality is not preserved from the two-dimensional space into the one-dimensional space with respect to R. This means that this mapping is not an optimal locality-preserving mapping for each individual point in the multi-dimensional space. In Section 4, we will define how a locality preserving mapping algorithm can be optimal with respect to all the points in the space.

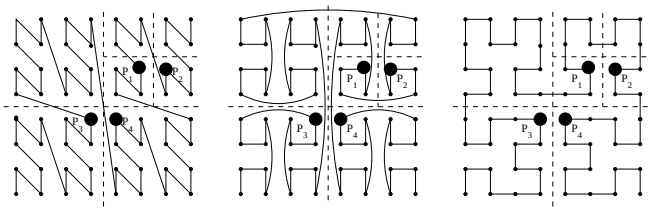


Fig. 2. The Fractal Locality-Preserving Mapping.

2.2 The Fractal Mapping

For the past two decades, recursive space-filling curves, which are special cases of fractals [34], have been considered a natural method for locality-preserving mappings. Mandelbrot [34], the father of fractals, derived the term *fractal* from the Latin adjective *fractus*. The corresponding Latin verb *frangere* means "to break" or "to fragment". Thus, fractals divide the space into a number of fragments, visiting the fragments in a specific order. Once a fractal starts to visit points from a certain fragment, no other fragment is visited until the current one is completely exhausted. By dealing with one fragment at a time, fractal locality-preserving mapping algorithms perform a local optimization based on the current fragment.

Local optimization is the major drawback in fractal locality-preserving mapping algorithms. Consider the case of two points P_i and P_j that lie on the boundaries of two different fragments and $|P_i - P_j| = 1$. Although P_i and P_j are near to each other in the multi-dimensional space, they will be far from each other in the one-dimensional space because they lie in different fragments. Figure 2 gives an example of this boundary effect on three different fractal locality-preserving mapping algorithms, the Peano, Gray, and Hilbert space-filling curves. In each curve, the space is divided into four quadrants. Each quadrant is divided recursively into another four quadrants, as in the upper right quadrant. Notice that, in these fractals, a quadrant represents a fragment of the space. For points P_1 and P_2 in Figure 2, although $|P_1 - P_2| = 1$ in the two-dimensional space, the distance between P_1 and P_2 in the one-dimensional space will be $6, 5, 11$ for the fractal mapping algorithms based on the Peano, Gray, and Hilbert space-filling curves, respectively. Things become even worse if we consider a finer resolution. For example, consider points P_3 and P_4 in Figure 2. $|P_3 - P_4| = 1$ in the two-dimensional space. The two points lie in two different quadrants and are far from each other in the one-dimensional space. The distance between P_3 and P_4 in the one-dimensional space will be $22, 47, 43$ if we use the mapping algorithms based on the Peano, Gray, and Hilbert space-filling curves, respectively.

The boundary effect problem in fractal locality-preserving mapping algorithms is unavoidable, and results in non-deterministic results. Any application that uses a locality-preserving mapping algorithm would expect to have the same

performance in preserving the locality for all multi-dimensional points. Fractal mapping algorithms favor the points that lie far from fragment borders. Points that lie near to the fragment borders fare the worst. In Section 6 we show how this property affects the performance of the fractal mapping algorithms.

[33, 44] address the boundary problem in fractals by using more than one space-filling curve. In [33], multiple shifted copies of the data are stored, where each copy is ordered by the Hilbert space-filling curve. In this case, if two points lie on the boundary of one copy of the data, then they will not lie on the boundary in the shifted copy. An algorithm for similarity search queries would search in all the shifted copies. A similar idea is proposed in [44], where multiple different space-filling curves are used for the same set of data. In this case, the set of candidate nearest neighbors is formed from the union of neighbors in accordance with the different space-filling curves. As can be observed, these are all heuristics for preserving the locality using fractals.

2.3 The Spectral Mapping

In this paper, we propose the use of the Spectral LPM, an optimal locality-preserving mapping with respect to all data points, to support multi-dimensional range queries and nearest-neighbor queries. The optimality proof is presented in Section 4. Spectral LPM avoids the drawbacks of the fractal algorithms by using a global optimization instead of a local one, where global optimization means that all multi-dimensional data points are taken into account when performing the mapping. Notice that the local optimization in fractals is achieved by considering only the points in the current fragment during the mapping process. Unlike fractals, Spectral LPM does not favor any set of points over the others; all points are treated in a similar way.

In general, spectral algorithms use the eigenvalues and eigenvectors of the matrix representation of a graph. Spectral algorithms are based on the spectral theory which relates a matrix to its eigenvalues and eigenvectors [46]. Spectral theory is attributed to David Hilbert, from a series of six papers collected and published as one volume in [23]. Although spectral algorithms are well known for more than 90 years, their use in the computer science fields began in [9], where the eigenvectors of the adjacency matrix $A(G)$ of a graph G are used in graph partitioning. A milestone in spectral algorithms is due to Fiedler [15, 16] who proposed using the eigenvalues and eigenvectors of the *Laplacian* matrix $L(G)$ of a graph G instead of the adjacency matrix $A(G)$. Following Fiedler's work, all spectral algorithms turn out to use the *Laplacian* matrix $L(G)$. Spectral algorithms have been widely used in graph partitioning [40, 41], data clustering [29], linear labeling of a graph [27], and load balancing [21]. The optimality of the spectral order in many applications is discussed in [6, 19, 27, 29, 45]. To the authors' knowledge, the use of a spectral mapping in database systems to support range and similarity search queries is novel.

P	A set of multi-dimensional points where $\|P\| = n$.
S	A linear order of all the points in P.
$G(V, E)$	A graph G with undirected edges E and vertices V, where $\|V\| = n$.
d_i	The degree of vertex $v_i \in V$.
$A(G)$	The *adjacency* matrix of G where $A(G)_{ii} = 0$, and $A(G)_{ij} = 1$ if the edge $(i, j) \in E$, *o.w* $A(G)_{ij} = 0$.
$D(G)$	The *diagonal* matrix of G where $D(G)_{ii} = d_i$, and $D(G)_{ij} = 0, \forall i \neq j$.
$L(G)$	The *Laplacian* matrix of G where $L(G)_{ii} = d_i$, and $L(G)_{ij} = -1$ if $(i, j) \in E$, *o.w* $L(G)_{ij} = 0$. For any graph G, $L(G) = D(G) - A(G)$.
λ_2	The second smallest eigenvalue for $L(G)$.
X_2	The eigenvector $(x_1, x_2, \cdots, x_n)$ that corresponds to the eigenvalue λ_2 (Also known as the Fiedler vector [16]).
e	The unary vector $(1, 1 \cdots, 1)$.

Table 1. Symbols used in the paper.

Algorithm Spectral LPM
Input: *A set of multi-dimensional points P.*
Output: *A linear order S of the set P.*

1. *Model the set of multi-dimensional points P as a graph $G(V, E)$ such that each point $P_i \in P$ is represented by a vertex $v_i \in V$, and there is an edge $(v_i, v_j) \in E$ if and only if $|P_i - P_j| = 1$.*
2. *Compute the graph Laplacian matrix $L(G) = D(G) - A(G)$.*
3. *Compute the second smallest eigenvalue λ_2 and its corresponding eigenvector X_2 of $L(G)$.*
4. *For each $i = 1 \rightarrow n$, assign the value x_i to v_i and hence to P_i*
5. *The linear order S of P is the order of the assigned values of P_i's.*
6. **return** S.
7. **End.**

Fig. 3. Pseudo code for the Spectral LPM.

3 The Spectral Mapping Algorithm

For the remainder of this paper, we use the notations and definitions given in Table 1. Based on this notation, the pseudo code for the Spectral LPM algorithm is given in Figure 3.

Figure 4 gives an example of applying Spectral LPM to a set of two-dimensional points in a 3×3 grid. The main idea of Spectral LPM is to model the multi-dimensional points as a set of vertices V in an undirected, unweighted graph $G(V, E)$ with an edge $e \in E$ between two vertices if and only if the two vertices represent two points with Manhattan distance $M = 1$. Figure 4b gives the graph modeling of the multi-dimensional points in Figure 4a. Then, the *Laplacian* matrix is computed as in Figure 4c, where row i in $L(G)$ represents vertex $v_i \in V$. Note that the order used in numbering the vertices in G in Figure 4b is not important. Different orders result in different permutations of the rows in $L(G)$, which will yield the same result. The main step in Spectral LPM is to compute the second smallest eigenvalue λ_2, also known as the *algebraic connectivity* of the graph [15], and its corresponding eigenvector X_2, also known

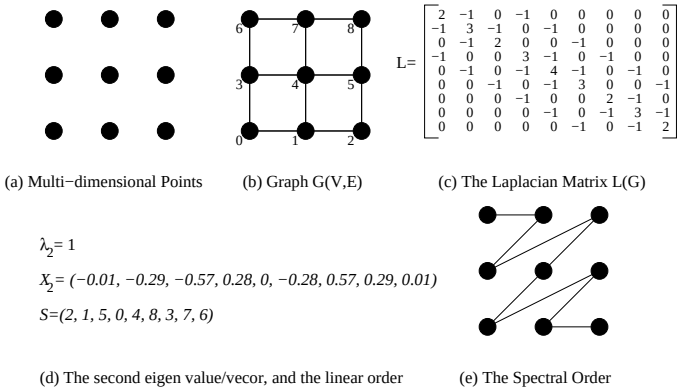


Fig. 4. The Spectral LPM algorithm.

as the *characteristic valuation* of the graph [15] or Fiedler vector [16]. Figure 4d gives λ_2 and X_2 for $L(G)$ in Figure 4c. The eigenvalues and eigenvectors of a matrix can be determined by any of the well known general iterative methods, e.g., [8, 31]. More specific numerical methods to compute the Fiedler vector are proposed in [24, 30]. For a survey on iterative methods for computing eigenvalues, the reader is referred to [18]. Finally, the value $x_i \in X_2$ is assigned to each vertex $v_i \in V$ and point $p_i \in P$. The spectral order S is determined by ordering the vertices and hence the points according to their assigned values, as in Figures 4d and 4e.

4 The Optimality of the Spectral Mapping

An optimal mapping preserves the locality from the multi-dimensional space into the one-dimensional space. In Section 2, we showed how a mapping can be optimal with respect to a given point in the multi-dimensional space, and we called such mapping a *local* optimal mapping. We showed in Figure 1 that we can not have such *local* optimal mapping for each individual point in the multi-dimensional space. In this section, we show how a mapping can be considered *globally* optimal for all points in the space. Then, we prove that Spectral LPM achieves this optimality.

Definition 1. : *A vector* $X = (x_1, x_2, \ldots, x_n)$ *that represents the* n *one-dimensional values of* n *multi-dimensional points represented as a graph* $G(V, E)$ *is considered to provide the globally optimal locality-preserving mapping from the multi-dimensional space into the one-dimensional space if* X *satisfies the following optimization problem:*

$$Minimize f = \sum_{(v_i,v_j)\in E} (x_i - x_j)^2 \qquad S.t: \qquad \sum_{i=1}^{i=n} x_i^2 = 1, \sum_{i=1}^{i=n} x_i = 0 \qquad (1)$$

As in Figure 4, the locality-preserving mapping problem from the multi-dimensional space into the one-dimensional space is the same as the problem of embedding a graph G into a line L. A globally optimal mapping maps any two vertices $v_i, v_j \in V$ where $(v_i, v_j) \in E$ to the points x_i, x_j respectively such that $|x_i - x_j|$ is minimized. In other words, the points with Manhattan distance $M = 1$ in the multi-dimensional space are required to be near to each other in the one-dimensional space. By making this concept global over all the edges in the graph (recall that there is an edge in the graph between each pair of points with $M = 1$), we obtain the objective function: $Minimize f = \sum_{(v_i,v_j)\in E} |x_i - x_j|$. To avoid the absolute operation, we use the square for the difference between any two points. The objective function becomes $Minimize f = \sum_{(v_i,v_j)\in E} (x_i - x_j)^2$. However, the minimization problem is invariant under translation, yielding an infinite number of solutions for X. For example, the vectors X and $X + a$ give the same result for f. To force a unique solution to the optimization problem, we pick any valid solution X' and apply a transformation by $a = -Average(X')$. Thus, the mapping vector X would be $X = X' - Average(X')$. The constraint $\sum_{i=1}^{i=n} x_i = 0$ forces a choice for X such that $\sum_{i=1}^{i=n} x_i = \sum_{i=1}^{i=n} x_i' - \sum_{i=1}^{i=n} Average(X') = \sum_{i=1}^{i=n} x_i' - n(\sum_{i=1}^{i=n} x_i'/n) = 0$. Another problem for the objective function is the existence of a trivial solution where all x_i's are set to 0, and hence f will be 0. To find a non-trivial solution, we normalize X by dividing all x_i's by $\| x \| = \sqrt{x_1^2 + x_n^2 + \cdots + x_n^2}$. With this normalization, an additional constraint is added where $\sum_{i=1}^{i=n} x_i^2 = 1$.

Theorem 1. : *The optimization problem in Definition 1 is equivalent to the following optimization problem:*

$$Minimize f = X^T L X \qquad S.t: \qquad X^T X = 1, X^T e = 0 \tag{2}$$

Proof. From the definition of the Laplacian matrix, we have $L(G) = D(G) - A(G)$. Therefore, the objective function can be rewritten as follows:

$$X^T L X = X^T D X - X^T A X \tag{3}$$

However, $X^T D X$ and $X^T A X$ can be rewritten in terms of x_i's as follows:

$$\begin{aligned} X^T D X &= X^T \begin{pmatrix} d_1 & 0 & \cdots & 0 \\ 0 & d_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & d_n \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} x_1 & x_2 & \cdots & x_n \end{pmatrix} \begin{pmatrix} d_1 x_1 \\ d_2 x_2 \\ \vdots \\ d_n x_n \end{pmatrix} \\ &= \sum_{i=1}^{n} d_i x_i^2 = \sum_{(v_i,v_j)\in E} (x_i^2 + x_j^2) \end{aligned} \tag{4}$$

and

$$X^T AX = \begin{pmatrix} x_1 & x_2 & \cdots & x_n \end{pmatrix} \begin{pmatrix} \sum_{(v_1,v_j)\in E} x_j \\ \sum_{(v_2,v_j)\in E} x_j \\ \vdots \\ \sum_{(v_n,v_j)\in E} x_j \end{pmatrix} = \sum_{i=1}^{n} x_i \sum_{(v_i,v_j)\in E} x_j = 2 \sum_{(v_i,v_j)\in E} x_i x_j \tag{5}$$

substituting from 4, 5 into 3 results in:

$$X^T LX = \sum_{(v_i,v_j)\in E} (x_i^2 + x_j^2) - 2 \sum_{(v_i,v_j)\in E} x_i x_j = \sum_{(v_i,v_j)\in E} (x_i - x_j)^2$$

So, the objective function $f = \sum_{(v_i,v_j)\in E}(x_i - x_j)^2$ is equivalent to the objective function $f = X^T LX$. The two constraints $X^T X = 1$ and $X^T e = 1$ are just the vector form representation for the two constraints $\sum_{i=1}^{i=n} x_i^2 = 1$, $\sum_{i=1}^{i=n} x_i = 0$, respectively. Proving that these constraints are equivalent is trivial.

Theorem 2. *[15]: The solution of the optimization problem in Theorem 1 is the second smallest eigenvalue λ_2 and its corresponding eigenvector X_2.*

Proof. Given the first constraint, the objective function can be rewritten as $X^T LX = X^T Xf = X^T fX \Rightarrow LX = fX$. Thus, X must be an eigenvector of L, with the corresponding eigenvalue f. Then, the solution of the optimization problem in Theorem 1 is the least non-trivial eigenvalue f, and its corresponding eigenvector X. Note that the eigenvector X is guaranteed to satisfy both constraints of the optimization problem with the transformation and normalization procedure discussed in Theorem 1. According to the Perron-Frobenius Theorem [17] , there is only **one maximum eigenvalue** for any non-negative irreducible[1] matrix M, which is $\rho(M)$ and is called the spectral radius of M. $\rho(M)$ is bounded by the minimum and maximum sum of all the rows in M. Applying this theorem on the non-negative irreducible matrix $M = (n-1)I - L(G)$, yields that $\rho(M) = n - 1$. Since, $\rho((n-1)I) = n - 1$, so $\rho(-L(G)) = 0$. This means that the matrix $L(G)$ has only **one minimum eigenvalue** with value 0, and therefore there is only one trivial eigenvalue for $L(G)$ [1]. This means that the first non-trivial eigenvalue for $L(G)$ is the second one. Thus, the minimization of $X^T LX$ is λ_2, the second smallest eigenvalue of L and its corresponding eigenvector X_2.

From Theorem 2, the eigenvector X_2 of the second smallest eigenvalue λ_2 (Step 3 in Spectral LPM) is the optimal solution of the optimization problems for Definition 1 and Theorem 1. Since the optimization problem in Definition 1 is modeled in the first step in Spectral LPM, then Spectral LPM guarantees the optimal result.

[1] A matrix M is irreducible iff it represents a connected graph.

5 Extensibility of the Spectral Mapping

Additional requirements cannot be integrated in the fractal locality-preserving mapping algorithms. For example, assume that we need to map points in the multi-dimensional space into disk pages, and we know (from experience) that whenever point x in page P_x is accessed, there is a very high probability that point y in page P_y will be accessed soon afterwards. Assume that x and y lie very far from each other in the multi-dimensional space. A consistent mapping would result in pages P_x and P_y being far away from each other on disk. However, it is clear that we need to map x and y into nearby locations in the one-dimensional storage (disk), fractals cannot help with such an additional requirement (i.e., the requirement of taking the probability of access into consideration). Fractals deal only with the location of the multi-dimensional points in the space. In contrast, Spectral LPM provides an extensible environment that can incorporate any number of additional requirements. The flexibility of Spectral LPM comes from the degree of freedom it has in Step 1 of the Spectral LPM Algorithm, given in Figure 3. Step 1 is the graph modeling, where any requirement can be modeled as an edge in the graph G in Step 1 of Spectral LPM. Returning to the example of two disk pages P_x and P_y. To force mapping x and y into nearby locations in the one-dimensional space using Spectral LPM, we add an edge (x, y) to the graph G. By adding this edge, Spectral LPM learns that x and y need to be treated as if they have Manhattan distance $M = 1$ in the multi-dimensional space.

Another extensibility feature in Spectral LPM is that we can change the way we construct the graph G. For example, we can model the multi-dimensional points in a graph G such that there is an edge between any two points P_i and P_j if and only if the maximum distance over any dimension is one. In case of the two-dimensional space, this results in an eight-connectivity graph where each point P_i is connected to its eight neighbors (compare with the four-connectivity graph in Figure 4b). Figure 5 gives the modeling of two-dimensional points in a 4×4 grid with the resulting spectral order after applying Spectral LPM for four-connectivity (Figures 5a, 5b) and eight-connectivity graphs (Figures 5c, 5d).

More generally, points in the multi-dimensional space can be modeled as a weighted graph, where the weight w of an edge $e(v_1, v_2)$ represents the priority of mapping v_1 and v_2 to nearby locations in the one-dimensional space. In this case, the definition of $L(G)$ will be changed slightly to have $L(G)_{ii} = \sum_{(i,j)\in E} w_{ij}$, and $L(G)_{ij} = -w_{ij}$ if $(i, j) \in E$, o.w., $L(G)_{ij} = 0$. Also, the objective function of Definition 1 will be $f = \sum_{(v_i,v_j)\in E} w_{ij}(x_i - x_j)^2$. However, Theorems 1 and 2 will be the same.

Notice that the proof of optimality of Spectral LPM in Section 4 is valid regardless of the graph type. The idea of Spectral LPM is that it is optimal for the chosen graph type. For the rest of the paper, we choose to work with the four-connectivity graph in Figure 4b where it has a very sparse and symmetric matrix that results in efficient computation time.

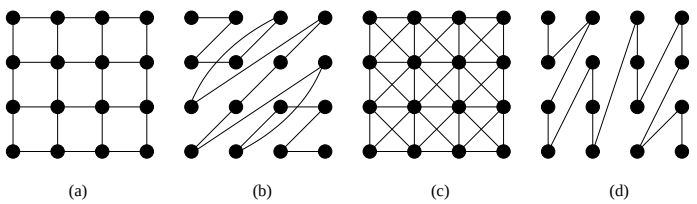


Fig. 5. Variation of the Spectral LPM algorithm. (a) Four-connectivity graph, (b) Its corresponding spectral order, (c) Eight-connectivity graph, (d) Its corresponding spectral order

6 Experimental Results

In this section, we give experimental evidence that Spectral LPM is superior to any of the fractal locality-preserving mapping algorithms. In the experiments, we focus on the effect of Spectral LPM on similarity search queries (e.g., nearest-neighbor queries) and range queries. However, due to its optimality, we believe that Spectral LPM will give similar superior performance when applied to other applications that require a locality-preserving mapping (e.g., the set of applications mentioned in Section 1).

We use the *Linear Span* for a given query selection (difference between the maximum and minimum linear coordinates in the selected region) as our measure of performance. The *Linear Span* measure is used in [26, 37, 38] to compare different fractal locality-preserving mappings. The lower the *Linear Span* of a given query, the better the locality-preserving mapping. The idea of the lower *Linear Span* is to have the ability to develop a single numeric index on a one-dimensional space for each point in a multi-dimensional space such that for any given object, the range of indices, from the smallest index to the largest, includes few points not in the object itself.

We evaluate Spectral LPM w.r.t. *Linear Span* by comparing Spectral LPM with three different fractal locality-preserving mapping algorithms based on the Peano, Gray, and Hilbert space-filling curves. In addition, we consider a row-major method as a simple and straightforward solution for mapping the multi-dimensional space into the one-dimensional space. In the experiments, we refer to the row-major mapping by the Sweep space-filling curve (SFC) [35]. The motivation for choosing the Sweep SFC is that it provides another way of multi-dimensional mapping that is not based on fractals. A popular example that uses the Sweep SFC is storing the multi-dimensional matrices in memory.

For the implementation of Spectral LPM, we use the *conjugate gradient method* [30] to compute the Fiedler vector of $L(G)$. The *conjugate gradient method* is proved to have less iterations and efficient time processing over other algorithms. In addition, the conjugate gradient method directly gives the eigenvector associated with the second smallest eigenvalue (the Fiedler vector) without the need to compute any other eigenvectors. For the Hilbert SFC, we use the methodology in [4] to generate the Hilbert SFC for an arbitrary number of

dimensions. The Peano and Gray SFCs can be easily implemented for the D-dimensional space as in [14]. The implementation of the Sweep SFC is straightforward.

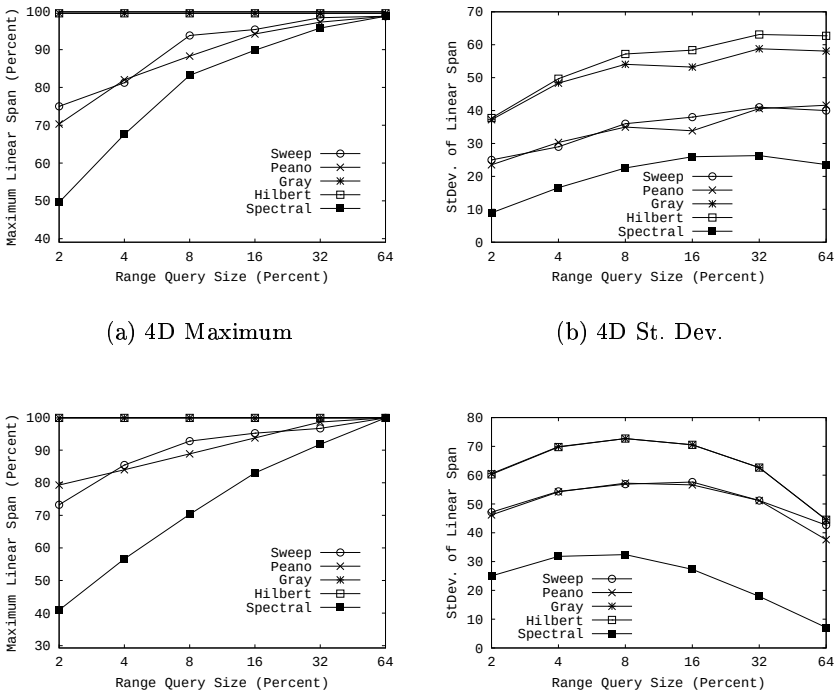
We perform two sets of experiments; mesh-data and real-data experiments. In the mesh-data experiments, we assume that there exist a data point at every grid cell in the multi-dimensional space, and we exhaustively test all possible range and nearest-neighbor queries. In the real-data experiments, we use a real data set and generate a batch of range queries and nearest-neighbor queries. Both the mesh-data and real-data experiments show the superior performance of the Spectral LPM over any other locality-preserving mappings.

6.1 Range Query Performance using Mesh-data

In the first set of experiments, we run all possible four-dimensional range queries with sizes ranging from 2% to 64% of the multi-dimensional space. Figure 6a gives the maximum possible *Linear Span* of range queries. Spectral LPM outperforms all other locality-preserving mappings, while the Gray and Hilbert SFCs give the worst performance. For example, for a query that retrieves only 2% of the multi-dimensional space, in the worst case, the Gray and Hilbert SFCs can map this query to span 100% of the one-dimensional space. Although, the boundary effect in fractals is the main reason behind this bad performance, it does not have the same bad effect on the Peano SFC. The main reason is that the Gray and Hilbert SFCs visit the space fragments in the order imposed by the gray code while the Peano SFC visits the space fragments in the order imposed by the binary code. Spectral LPM has the smallest Linear Span. This demonstrates the notion of global optimality that Spectral LPM has. In other words, Spectral LPM optimizes over the entire space and treats the multi-dimensional space uniformly, and hence its worst-case Linear Span is much smaller than the other SFCs.

Figure 6b tests the stability of the locality-preserving mapping. A good locality-preserving mapping should provide the same performance for each query size, regardless of its location in the space. The standard deviation of the *Linear Span* is used as a measure of the stability of the locality-preserving mapping. Lower standard deviation indicates more stability. As expected (due to the boundary effect), the Gray and Hilbert SFCs gives the worst performance. Spectral LPM outperforms all other mappings for all range query sizes. The Peano and Sweep SFCs give an intermediate performance. Notice that although the Sweep SFC is not a fractal, it gives the same performance as the Peano fractal mapping. The main reason is that the Sweep SFC discriminates between the dimensions. For example, in the two-dimensional Sweep SFC, a range query that asks for all points with y=1 would result in an excellent performance, while the query that asks for all points with $x = 1$ would result in a very bad performance. For all cases, the Spectral LPM does not suffer from discriminating between dimensions, or boundary effect.

The same results are obtained when we perform the same experiments in the five-dimensional space. In general, the relative performance of the Spectral



(a) 4D Maximum (b) 4D St. Dev.

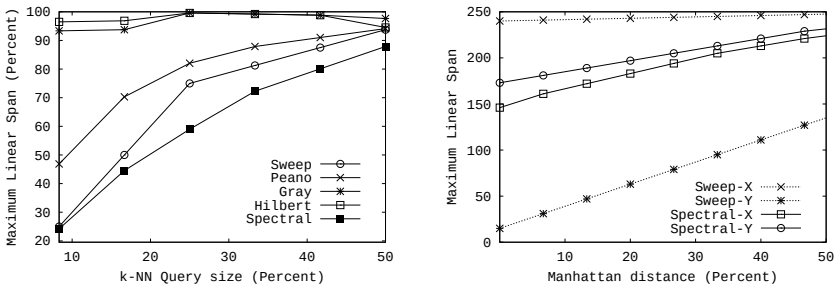
(c) 5D Maximum (d) 5D St. Dev.

Fig. 6. The performance of range queries in 4D, 5D spaces.

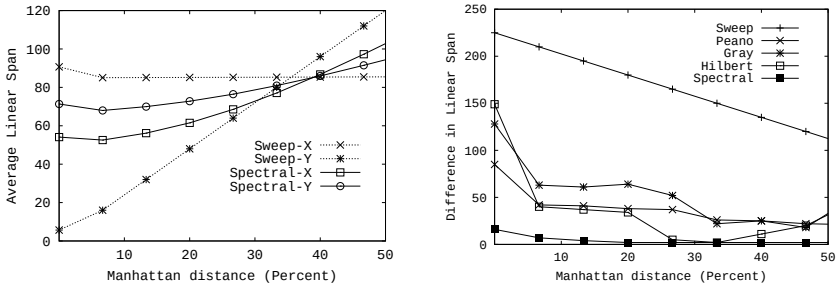
LPM over other mappings increases with the space dimensionality. Figures 6c, and 6d give the maximum and standard deviation of the *Linear Span* in the five-dimensional space, respectively. Notice that the standard deviation of the *Linear Span* decreases with large query sizes for all locality-preserving mappings. This can be clarified if we consider the extreme case of a range query that covers 100% of the space. Clearly, there is only one range query with 100% space coverages. This results in zero standard deviation.

6.2 k-Nearest-Neighbor Using Mesh-data

Figure 7 gives the performance of k-nearest-neighbor (k-NN) queries. In the case of mesh data, we have data points in all space points. As a result, in the two-dimensional space, when setting $k = 4$, k-NN retrieves the four neighbors with Manhattan distance 1. Figures 7a gives the maximum *Linear Span* of all possible k-nearest-neighbor queries with query size up to 50% of the four-dimensional space. The spectral mapping gives much better performance than fractals with respect to the maximum linear span.



(a) 4D Maximum (b) 2D Maximum



(c) 2D Average (d) 2D Difference

Fig. 7. The performance of k-NN queries in 2D and 4D spaces using mesh data.

Since Spectral LPM and the Sweep SFC have the best performance, in Figures 7b and 7c we compare the performance of the Spectral LPM and the Sweep SFC with respect to different space dimensions. For simplicity in presenting the results, the experiment is performed only for the two-dimensional space. The x axis represents the Manhattan distance over only one dimension. The y axis represents the maximum possible *Linear Span* in the one-dimensional space for every two points with a certain Manhattan distance up to 50% of the two-dimensional space. By the curves Sweep-X and Sweep-Y, we mean that we compute the Manhattan distance over the X and Y dimensions, respectively. The same argument is valid for Spectral-X and Spectral-Y. The performance of the Sweep mapping have much variation when measuring the distance over the X (Sweep-X) and Y (Sweep-Y) dimensions. However, for the Spectral mapping, the performance is very similar for the two dimensions. For example, a query that asks for a point Q that have similar y value as point P ($M = 0$) would guarantee to have an answer that have one-dimensional distance at most 15 (Figure 7b) with average 6 (Figure 7c). However, if the same query asks for a point Q that has similar x value, instead of y, then the answer would have one-dimensional distance that is up to

240 with average 91. On the other side, Spectral LPM answers the first query in one-dimensional distance up to 146 with average 54 and the second query in a one-dimensional distance that is up to 173 with average 71. The high variation of the Sweep mapping makes it non-deterministic and favors some queries over the others. Such high variation is not desirable by any locality-preserving mapping.

Figure 7d performs the same experiment for all locality-preserving mappings. The plotted curves represent the difference in the maximum one-dimensional distance that corresponds to Manhattan distance M for X and Y dimensions. The Sweep and Spectral curves can be derived by getting the absolute difference $|SweepY - SweepX|$ and $|SpectralY - SpectralX|$ from Figure 7b, respectively. The Sweep mapping gives very bad performance. Spectral LPM almost gives an optimal result, where the difference is almost 0. Fractals, have a moderate performance that is not as good as Spectral LPM nor as bad as the Sweep mapping.

6.3 Performance Using Real-data Sets

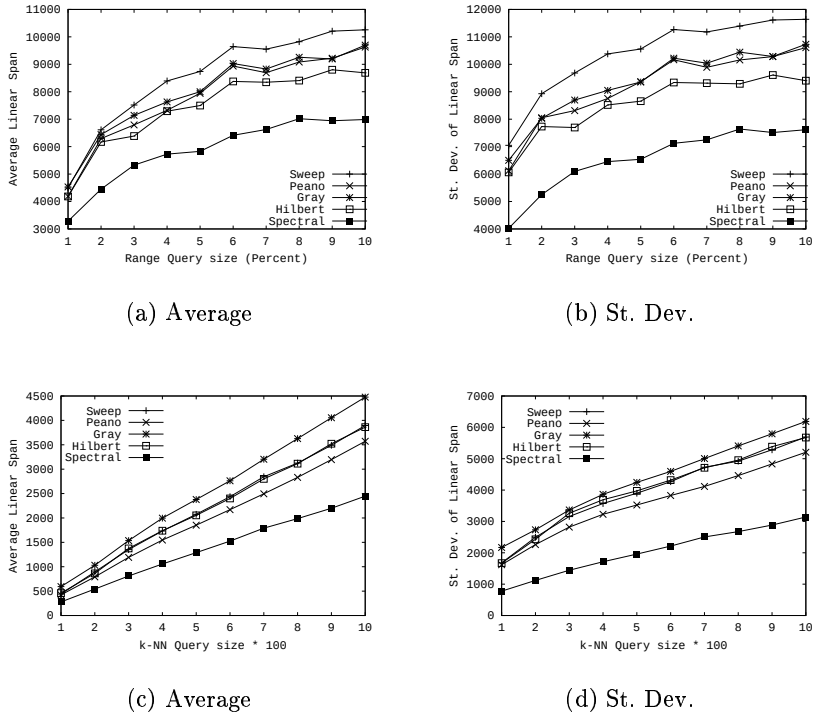
In this section, we use the North East data set that contains 123,593 postal addresses, which represent three metropolitan areas (New York, Philadelphia and Boston) [25]. The two-dimensional space is represented by a 128×128 grid. Each grid cell corresponds to a disk page. Data points are aligned to the nearest grid cell. Disk pages are stored in the order imposed by the underlying locality-preserving mapping. It is required that the locality-preserving mapping clusters the disk pages required to answer a specific query in a minimum *Linear Span*.

In the first experiment (refer to Figures 8a and 8b), we run 10,000 random range queries with sizes from 1% to 10% of the space. Figure 8a gives the average size of the *Linear Span* for each query size. Clearly, the Spectral LPM outperforms all other mappings. As the query size increases, the relative performance of the Spectral LPM over other mappings increases. Figure 8b measures the stability of the locality-preserving mappings with regard to the location of the range query. The standard deviation of the *Linear Span* of range queries is used as an indication for the stability. The Spectral LPM outperforms all other mappings.

In the next experiment, (refer to Figures 8c and 8d), we run 10,000 random k-nearest-neighbor queries with k ranges from 100 to 1000. Figures 8c and 8d give the average and standard deviation of the *Linear Span*, respectively. Again, the results from the real data set agrees with the analytical results that the Spectral LPM outperforms all other locality-preserving mappings.

7 Conclusion

In this paper, we argue against the use of fractals as a basis for locality-preserving mapping algorithms by providing some examples and experimental evidence to show how fractal mapping algorithms produce a poor mapping. Then, we introduce the Spectral LPM; a provably optimal algorithm for mapping the



(a) Average (b) St. Dev.

(c) Average (d) St. Dev.

Fig. 8. The size of *Linear Span* of (a),(b) Range Queries, (c),(d) k-NN Queries

multi-dimensional space into the one-dimensional space such that the points that are nearby in the multi-dimensional space would still be nearby in the one-dimensional space. Spectral LPM uses the spectral properties of the multi-dimensional space where the multi-dimensional points are mapped into a graph $G(V, E)$. Then, the linear order of the multi-dimensional points is determined by their order within the eigenvector X_2 that corresponds to the second smallest eigenvalue λ_2 of the *Laplacian* matrix $L(G)$. In addition, we provide a mathematical proof for the optimality of Spectral LPM. Unlike fractals, Spectral LPM can incorporate any number of additional requirements for the locality-preserving mapping. Experimental analysis confirms the superior performance of Spectral LPM over the long used fractal locality-preserving mapping algorithms for similarity search queries and range queries.

References

1. W. N. Anderson and T. D. Morley. Eigenvalues of the laplacian of a graph. Technical Report TR-71-45, University of Maryland, Oct. 1971. Reprinted in Linear and Multilinear Algebra, 18:141-145, 1985.

2. W. G. Aref, K. El-Bassyouni, I. Kamel, and M. F. Mokbel. Scalable qos-aware disk-scheduling. In *Intl. Database Engineering and Applications Symp., IDEAS*, Alberta, Canada, July 2002.
3. J. J. Bartholdi and L. K. Platzman. An o(n log n) traveling salesman heuristic based on space filling curves. *Operation Research Letters*, 1(4):121–125, Sept. 1982.
4. T. Bially. Space-filling curves: Their generation and their application to bandwidth reduction. *IEEE Transactions on Information Theory*, 15(6):658–664, Nov. 1969.
5. C. Bohm, G. Klump, and H.-P. Kriegel. *xz*-ordering: A space-filling curve for objects with spatial extension. In *Intl. Symp. on Advances in Spatial Databases, SSD*, pages 75–90, Hong Kong, July 1999.
6. T. F. Chan, P. Ciarlet, and W. K. Szeto. On the optimality of the median cut spectral bisection graph partitioning method. *SIAM Journal on Scientific Computing*, 18(3):943–948, May 1997.
7. D. Comer. The ubiquitous b-tree. *ACM Comp. Surveys*, 11(2):121–137, June 1979.
8. E. Davidson. The iterative calculation of a few of the lowest eigenvalues and corresponding eigenvectors of large real-symmetric matrices. *Journal of Computational Physics*, 17:87–94, 1975.
9. W. Donath and A. Hoffman. Algorithms for partitioning of graphs and computer logic based on eigenvectors of connection matrices. *IBM Technical Disclosure Bulletin*, 17:938–944, 1972.
10. C. Faloutsos. Multiattribute hashing using gray codes. In *Intl. Conf. on Management of Data, SIGMOD*, pages 227–238, Washington D.C., May 1986.
11. C. Faloutsos. Gray codes for partial match and range queries. *IEEE Transactions on Software Engineering, TSE*, 14(10):1381–1393, Oct. 1988.
12. C. Faloutsos and P. Bhagwat. Declustering using fractals. In *Intl. Conf. on Parallel and Distributed Information Sys.*, pages 18–25, San Jose, CA, Jan. 1993.
13. C. Faloutsos and Y. Rong. Dot: A spatial access method using fractals. In *Intl. Conf. on Data Engineering, ICDE*, pages 152–159, Japan, Apr. 1991.
14. C. Faloutsos and S. Roseman. Fractals for secondary key retrieval. In *Symp. on Principles of Database Systems, PODS*, pages 247–252, Mar. 1989.
15. M. Fiedler. Algebraic connectivity of graphs. *Czechoslovak Math. Journal*, 23(98):298–305, 1973.
16. M. Fiedler. A property of eigenvectors of nonnegative symmetric matrices and its application to graph theory. *Czechoslovak Math. Journal*, 25(100):619–633, 1975.
17. F. G. Frobenius. Uber matrizen aus nicht negativen elementen. *Sitzungsberichte der Koniglich Preusischen Akademie der Wissenschaften zu Berlin*, 4:456–477, 1912.
18. G. H. Golub and H. A. van der Vorst. Eigenvalue computation in the 20th century. *Jour. of Comp. and App. Math.*, 123(1-2):35–65, 2000.
19. S. Guattery and G. L. Miller. On the quality of spectral separators. *SIAM Journal on Matrix Analalysis and Applications*, 19(3):701–719, July 1998.
20. A. Guttman. R-trees: A dynamic index structure for spatial indexing. In *Intl. Conf. on Management of Data, SIGMOD*, pages 47–57, Boston, MA, June 1984.
21. B. Hendrickson and R. Leland. Multidimensional spectral load balancing. In *SIAM Conf. on Parallel Processing*, pages 953–961, 1993.
22. D. Hilbert. Ueber stetige abbildung einer linie auf ein flashenstuck. *Mathematishe Annalen*, pages 459–460, 1891.
23. D. Hilbert. *Grundzuge einer allgemeinen Theorie der linearen Integralgleinhungen.* Teubner, Leipzig, 1912.
24. M. Holzrichter and S. Oliveira. A graph based method for generating the fiedler vector of irregular problems. In *Parallel and Distributed Processing, LNCS*, volume 1586, pages 978–985. Springer Verlag, Apr. 1999.

25. http://dias.cti.gr/ ytheod/research/datasets/spatial.html.
26. H. V. Jagadish. Linear clustering of objects with multiple attributes. In *Intl. Conf. on Management of Data, SIGMOD*, pages 332–342, Atlantic City, NJ, June 1990.
27. M. Juvan and B. Mohar. Optimal linear labelings and eigenvalues of graphs. *Discrete Applied Mathematics*, 36:153–168, 1992.
28. I. Kamel and C. Faloutsos. Hilbert r-tree: An improved r-tree using fractals. In *Intl. Conf. on Very Large Databases, VLDB*, pages 500–509, Chile, Sept. 1994.
29. R. Kannan, S. Vempala, and A. Vetta. On clusterings - good, bad and spectral. In *Symp. on Foundations of Computer Science, FOCS*, pages 367–377, Redondo Beach, CA, Nov. 2000.
30. N. P. Kruyt. A conjugate gradient method for the spectral partitioning of graphs. *Parallel Computing*, 22(11):1493–1502, Jan. 1997.
31. C. Lanczos. An iteration method for the solution of the eigenvalue problem of linear differential and integral operators. *Journal of Research of the National Bureau of Standards*, 45(4):255–282, 1950.
32. J. K. Lawder and P. J. H. King. Querying multi-dimensional data indexed using the hilbert space filling curve. *SIGMOD Record*, 30(1), Mar. 2001.
33. S. Liao, M. A. Lopez, and S. Leutenegger. High dimensional similarity search with space-filling curves. In *Intl. Conf. on Data Engineering, ICDE*, pages 615–622, Heidelberg, Germany, Apr. 2001.
34. B. B. Mandelbrot. *Fractal Geometry of Nature*. W. H. Freeman, New York, 1977.
35. M. F. Mokbel and W. G. Aref. Irregularity in multi-dimensional space-filling curves with applications in multimedia databases. In *Intl. Conf. on Information and Knowledge Managemen, CIKM*, Atlanta, GA, Nov. 2001.
36. M. F. Mokbel, W. G. Aref, and A. Grama. Spectral lpm: An optimal locality-preserving mapping using the spectral (not fractal) order. In *Intl. Conf. on Data Engineering, ICDE*, pages 699–701, Bangalore, India, Mar. 2003.
37. B. Moon, H. Jagadish, C. Faloutsos, and J. Salz. Analysis of the clustering properties of hilbert space-filling curve. *IEEE Transactions on Knowledge and Data Engineering, TKDE*, 13(1):124–141, 2001.
38. J. A. Orenstein. Spatial query processing in an object-oriented database system. In *Intl. Conf. on Management of Data, SIGMOD*, pages 326–336, May 1986.
39. G. Peano. Sur une courbe qui remplit toute une air plaine. *Mathematishe Annalen*, 36:157–160, 1890.
40. A. Pothen. Graph partitioning algorithms with applications to scientific computing. *Parallel Numerical Algorithms*, 4(8):888–905, Jan. 1997.
41. D. Powers. Graph partitioning by eigenvectors. *Lin. Alg. Appl*, 101:121–133, 1988.
42. H. Sagan. *Space Filling Curves*. Springer, Berlin, 1994.
43. K. C. Sevcik and N. Koudas. Filter trees for managing spatial data over a range of size granularities. In *Intl. Conf. on Very Large Databases, VLDB*, pages 16–27, Bombay, India, Sept. 1996.
44. J. Shepherd, X. Zhu, and N. Megiddo. A fast indexing method for multidimensional nearest neighbor search. *SPIE, Storage and Retrieval for Image and Video Databases*, 3656:350–355, 1998.
45. H. D. Simon and S.-H. Teng. How good is recursive bisection. *SIAM Journal on Scientific Computing*, 18(5):1436–1445, Sept. 1997.
46. L. A. Steen. Highlights in the history of spectral theory. *American Math. Monthly*, 80(4):359–381, Apr. 1973.
47. I. Witten and M. Neal. Using peano curves for bilevel display of continuous tone images. *IEEE Computer Graphics and Applications*, pages 47–52, 1982.