Hilbert R-tree 構築における クラスタリング次元に関する研究

樋口 直哉[2]　デウィ ノールシッタ アジダ アブドゥル アジズ[1]　今村 安伸[2]　篠原 武[1]

概要：画像や音楽といった大量な多次元データを高速に近似検索するために R-tree や M-tree などの空間索引構造を用いるのが一般的である．R-tree は高次元データを扱うと検索効率が悪化するため，次元縮小射影 Simple-Map を用いる．R-tree の亜種である Hilbert R-tree は，本研究では，射影データの最初の一部の次元（クラスタリング次元という）だけを用いて順序付けする．順序付けしたデータに対して MBR で分割し，索引構造を構築する．本論文では，クラスタリングや MBR の次元数に関する最適化の検証を行う．実験により，従来手法と比べ，画像データにおいてはクラスタリングと MBR の次元数が射影次元数より小さくした場合が高速となることがわかった．

キーワード：近似検索，Hilbert R-tree，クラスタリング次元，MBR 次元，射影次元

A Study on Clustering Dimension in the Construction of Hilbert R-tree

Naoya Higuchi[2]　Dewi Noorsyitta Azida Abd Aziz[1]　Yasunobu Imamura[2]　Takeshi Shinohara[1]

Abstract: In order to perform similarity search processes for enormous amount of high-dimensional data in a high speed, generally, spatial indexes such as R-tree and M-tree are being used. R-tree's efficiency, however, is deteriorated when dealing with high-dimensional data and for that reason dimension reduction mapping Simple-Map is used. Hilbert R-tree is a type of R-tree which in our research, only a part of the projection dimension (called clustering dimension) is being used when arranging the data. The ordered data is then clustered in MBR before constructing the index structure. In this paper, we are verifying the optimization for the clustering and MBR dimension. From the experiments done on image data, compared to the conventional method, when the number of clustering and MBR dimension is decreased, the searching time becomes faster.

Keywords: similarity search，Hilbert R-tree，clustering dimension，MBR dimension，projection dimension

1. はじめに

近年，計算機の性能力の飛躍的な向上やインターネットの発達により，映像や音楽といったマルチメディアデータを大量に取り扱うことができるようになった．そのため，ユーザが必要とする情報を大量のデータの中から効率よく探し出す情報検索技術が重要となっている．

マルチメディアデータの検索においては，特徴データが質問点と完全に一致する情報のみを取得する完全一致検索よりも，似ているデータを探し出す近似検索のほうが重要である．その理由として，マルチメディアデータはデータ圧縮などの処理により質の劣化がおきるため，完全一致検索では，データの劣化や加工に対応することができず，人間の目から見て同じような画像でも計算機にとっては全く違うものとして扱われる場合が多く，ユーザーが目的とす

[1] 九州工業大学情報工学部
School of Computer Science and Systems Engineering, Kyushu Institute of Technology
[2] 九州工業大学情報工学府
Graduate School of Computer Science and Systems Engineering, Kyushu Institute of Technology

るデータを取得できなくなるからである．

膨大な量のマルチメディアデータを対象に高速な近似検索する手法として，階層的空間索引を用いて空間を索引付けする手法があり，B-tree [1] を多次元に拡張した R-tree [2] や M-tree [3] などの索引構造が一般的に用いられる．各々の索引構造に対しても様々な亜種が提案されている．例えば R-tree に注目すると，R$^+$tree[4] や Hilbert R-tree（HR-tree）[5] などがある．HR-tree は R-tree の亜種の中でも最も検索効率が良い索引構造の一つであることが過去の研究で確認されている．

HR-tree の特徴は，索引構造を構築する前に，登録する多次元データに対して空間充填曲線の一つである Hilbert 曲線 [6] を用いて一次元順序付けするという点である．そして，その順序付け通りにノードにデータを格納していき，ボトムアップに構築する．本研究では Hilbert 曲線に沿った順序付けを高速に行うために，田中が提案した手法であるヒルベルトソート [7] を順序付け手法として採用する．順序付けしたデータに対して，空間上のオブジェクトを包括する最小包囲矩形 （MBR: Minimum Bounding Rectangle）で分割し HR-tree を作成する．しかし，高次元特徴空間に対してそのまま HR-tree を構築すると次元の呪いにより検索の性能が悪化してしまう．そのため，特徴空間をより低次元に射影する必要があり，次元縮小射影法として Simple-Map（S-Map） [8] を用いる．

先行研究の従来手法としては，HR-tree 構築際のヒルベルトソート次元数と MBR 次元数に射影次元をすべて用いている．本研究では，射影データの最初の一部の次元（クラスタリング次元という）だけを用いてソートする．すなわち，本論文では，適切なクラスタリング次元数と MBR 次元数に関する最適化の検証を行うことが目的である．

本論文の構成は以下の通りである．近似検索と空間索引構造について 2 章で，クラスタリング次元について 3 章で述べる．4 章では検索の最適化のためにクラスタリングと MBR の次元数を変化させる実験を行い，それのまとめを 5 章で述べる．

2. 近似検索と索引構造

2.1 近似検索

近似検索とは質問点と近似するデータをデータベースから取り出すことである．データ間に非近似度（距離）を定義し，質問点からの距離の順番でデータを取り出すことにより，近似検索を実現する．近似検索システムは，質問点との距離が近いオブジェクトを**似ている**と判断し，それらを特徴空間から取り出すシステムのことである．近似検索のデータベースが対象とする特徴空間全体を $\mathcal{U} = \mathbb{R}^n$ とする．ここで，$\mathbb{R}$ は実数全体，n は特徴データの次元数である．任意の 2 点のオブジェクト間の非近似度の指標を示す**距離関数**を $d : \mathcal{U} \times \mathcal{U} \to \mathbb{R}^+$ とし，$\mathcal{D} = (\mathcal{U}, d)$ を距離空間とする．本論文で扱う近似検索では，距離関数 d は**距離の公理**である次の条件を満たすものとする．

(1) $d(X,Y) \geq 0$ (非負性)
(2) $d(X,Y) = d(Y,X)$ (対称性)
(3) $d(X,Y) \leq d(X,Z) + d(Z,Y)$ (三角不等式)
(4) $d(X,Y) = 0 \Leftrightarrow X = Y$ (同一性)

ここで，$X, Y, Z \in \mathcal{D}$ である．上の条件の中で最も重要なものは，三角不等式である．簡単のために，距離関数は単に**距離**と呼ぶこともある．

次に，近似検索で用いる距離計測について説明する．特徴空間内の任意のオブジェクトを x とする．x の特徴は n 個の実数の組 $(x_{(1)}, x_{(2)}, \ldots, x_{(n)})$ で表される．以下の三つの距離計測関数は距離の公理を満たすものである．

$$L_1\text{距離} : D(x,y) = \sum_{i=1}^{n} |x_{(i)} - y_{(i)}| \qquad (1)$$

$$L_2\text{距離} : D(x,y) = \sqrt{\sum_{i=1}^{n} (x_{(i)} - y_{(i)})^2} \qquad (2)$$

$$L_\infty\text{距離} : D(x,y) = \max_{i=1}^{n} |x_{(i)} - y_{(i)}| \qquad (3)$$

本論文では，元の空間での距離計算（実距離計算）を L_1距離，射影後の座標においての距離計算を L_∞距離 で行う．

2.2 質問方法

近似検索で用いる質問方法は，質問オブジェクトや要求する非近似度の範囲によって定義される．非近似度の範囲は，一般的に距離によって表される．質問に対する解は，その選択基準を満たすすべてのオブジェクトである．質問により得られた解の集合は順序付けされたリストと捉えることができ，順序付けの基準は質問点からの距離である．

近似検索では，主に範囲質問および近傍質問の 2 種類の質問方法が用いられている．質問点 q と距離 $r \in \mathbb{R}^+$ を質問のパラメータとする**範囲質問** $Range(\mathcal{D}, q, r)$ は，q から距離 r 以内のオブジェクトを取得する質問である．すなわち，その解の集合は，データベース内のオブジェクト全体を $\mathcal{S} \subseteq \mathcal{U}$ とすると，

$$Range(\mathcal{D}, q, r) = \{o \in \mathcal{S} \mid d(o,q) \leq r\}$$

である．それに対して，質問点 q から距離が最も小さいオブジェクトを取得する最近傍質問 $NN(\mathcal{D}, q)$ がある．すなわち，その解の集合は，

$$NN(\mathcal{D}, q) = \{o \in \mathcal{S} \mid o \text{ は } d(o,q) \text{ が最小のもの } \}$$

である．本実験では、上記二つのうちの最近傍質問を用いて近似検索を行う．

最近傍質問では初期検索範囲 r を無限大に設定する．そこから検索をはじめて，検索範囲内のオブジェクトを見つけた場合，そのオブジェクトを暫定解として登録し，検索範囲 r を暫定解と質問点との距離に収縮させる．以上の操作を検索範囲内に新たなオブジェクトが見つからなくなるまで繰り返す．

2.3 R-tree

R-tree はオブジェクトの追加・削除などの動的な操作を可能とする多岐平衡木である．R-tree は，対象とする空間を n 次元超直方体である最小包囲矩形 (MBR) で分割する．n はオブジェクトの次元数である．空間的に距離の近いオブジェクト同士ができるだけ同じノード内に格納されるように MBR を構成する．R-tree の内部ノードはそのすべての子ノードを包括する MBR と子ノードへのポインタを持ち，葉ノードにはデータベース内のオブジェクトへのポインタを格納する．ここで，互いに異なるノードを包括する MBR 同士が重なりあう可能性があることに注意する必要がある [9], [10]．高次元の特徴空間に対して R-tree を構築する場合，検索効率が悪化することが知られている．このように，特徴空間の次元が高くなるにつれて MBR の形状が正規形から離れたり，重なりが大きくなったりすることで検索効率が悪化する問題のことを**次元の呪い**と呼ぶ．本論文では，高次元空間での検索性能の悪化を緩和するために，次元縮小法である S-Map を用いて，高次元の特徴空間を低次元空間へ射影する．

R-tree における検索では，MBR と質問点との距離の順に，MBR と質問範囲が交差しているノードのみを訪問する．次元縮小を用いて R-tree を構築した場合には，葉ノードにおいて，オブジェクトと質問点との距離計算を行う前に，射影空間上の距離を調べることで，実距離計算回数を最小限に抑え，効率的な検索を実現する．検索は根ノードから始まり，深さ優先探索でノードを探索していく．検索が葉ノードのとき，その葉ノードが保持しているオブジェクトとの距離を計算し，検索範囲よりも小さければ検索範囲を更新する．検索が内部ノードのとき，質問点とその子ノードの MBR との距離を計算し，検索範囲と交差する MBR を持つノードのみ訪問する．この時，訪問する必要のあるノードは Active Branch List（ABL）と呼ばれる優先度付き待ち行列に挿入する．ABL 内のノードは，質問点と MBR との距離の昇順でソートされる．ABL の先頭から順にノードを訪問することで，質問点との距離が小さい MBR を持つノードから訪問することが可能になる．このようにノードの訪問順を制御することで，質問点の検索範囲の収縮を早めることができる．

図 1 には MBR A，B，C と質問点を示している．例えば，図 1 のように全ての MBR A，B，C が初期検索範囲と交差している状況を考える．訪問順を考慮しない場合は要素順，この例では MBR A，B，C の順で全て訪問する．これに対し，訪問順を制御する場合は質問点から最も近い MBR B から訪問し，MBR B に含まれるオブジェクトのうち質問点と最も近いオブジェクトとの距離まで検索範囲を収縮する．これにより，訪問順を制御しなかった場合には訪問しなければならなかった MBR A を訪問する必要がなくなる．このように，R-tree は訪問順を制御することで検索範囲の収縮を早め，効率的な検索を実現している．

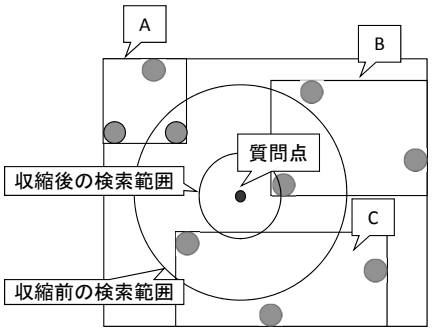

図 1 R-tree による検索
Fig. 1 Similarity search by R-tree

また，R-tree には様々な亜種が提案されている．しかし，その中でも HR-tree は，空間充填曲線である Hilbert 曲線を用いて構築されるため，通常の R-tree に比べて検索効率が良いことが知られている．この HR-tree については 3 章で詳しく説明する．

3. Hilbert R-tree とクラスタリング次元

多次元空間内の全ての格オブジェクトをただ一度だけ通る曲線を，空間充填曲線と呼ぶ．空間内でこの曲線が空間内で通った順番にオブジェクトを順序付けすることで，多次元空間上のオブジェクトを一次元順序付けすることが可能となる．空間充填曲線の中でも Hilbert 曲線は，多次元データを空間的に近い順に一次元順序付けできることが知られている．過去の研究で，R-tree を Hilbert 曲線を用いて構築した HR-tree は，R-tree に比べ検索効率が大幅に向上することが確認されている．

Hilbert 曲線順序付け手法としては，Hilbert 値と呼ばれる Hilbert 曲線上の始点からの距離を用いるのが一般的である．しかし，Hilbert 値を得るためには，空間を分割していく必要があり，空間の次元が上がるほど指数的にコストが増大してしまう．そのため，Hilbert 値を割り振って順序付けを行う手法では，次元が上がるほどに処理時間が極端に遅くなってしまうという問題がある．この問題を解決するために，過去の研究での田中はヒルベルトソートという手法を提案した．ヒルベルトソートにおいては，オブジェクトが一つしか残らないおよび存在しない部分空間をそれ以上分割せず，空間の分割と共にオブジェクトの順序

付けを行うという特徴を持つ．そのためヒルベルトソートでは，Hilbert 値を求めることなく Hilbert 曲線順序付けを行うことができ，次元が上がっても一定時間で処理が可能である．ヒルベルトソートを用いることで高速に Hilbert 曲線順序付けを行うことができる．

今までの研究では，特徴空間を S-Map で任意の低次元空間に射影し，射影した空間を HR-tree で索引する．上記で述べたように，登録する多次元データに対してヒルベルトソートで順序付けを行い，その順番で一定数（ノードサイズという）ごとにデータをまとめたクラスタを葉ノードに格納していく．そして葉ノードから根ノードへ向けてボトムアップに索引構造を構築する．先行研究では，HR-tree 構築際のヒルベルトソート次元数と MBR 次元数は，低次元に射影した S-Map の射影次元をすべて用いているが，本研究では，射影データの最初の一部だけを用いてソートする．その一部の次元を**クラスタリング次元**と呼ぶ．

4. 実験

実験に用いるのは画像データである．画像データは約 2,800 本の動画から切り出した約 700 万件の画像フレームを 64 次元に特徴抽出し登録したデータベースである．質問データにはデータベース内によく似たデータがある**近質問**，やや似たデータがある**準近質問**，似たデータがない**遠質問**を約 3 万件ずつ用い，計約 9 万件である．質問データに関しては実験時間短縮のため，質問を 100 飛ばしで計約 900 件で行う．質問方法は最近傍質問を行う．データを S-Map で低次元射影して，以下の二つの実験を行い従来手法との比較検証を行う．

- 実験 1：クラスタリング次元数減少
- 実験 2：クラスタリング＋ MBR の次元数減少

本実験では，以下の 5 つの射影次元とノードサイズの場合に関して調べる．

(1) 8 次元射影ノードサイズ 900
(2) 12 次元射影ノードサイズ 800
(3) 16 次元射影ノードサイズ 800
(4) 20 次元射影ノードサイズ 600
(5) 32 次元射影ノードサイズ 600

また，実験に用いた PC は表 1 の通りである．

表 1 実験に用いた PC の性能
Table 1 The SPEC of the PC used in Experiment

CPU	Intel(R) Core(TM) i7-3770 3.40GHz
メモリ	16GBytes

4.1 実験 1（クラスタリング次元の影響）

クラスタリング次元を射影次元数より減少させて実験を行った．（1）～（5）に関してそれぞれの実験結果を表 2，表 3，表 4，表 5，表 6 に示す．訪問ノード数，距離計算回数，検索時間はそれぞれ 1 質問当たりの平均値であり，太字はそれぞれの最小値を表している．

結果として，検索時間に関しては（1）～（5）いずれの場合においても従来手法と比べて，射影データの最初一部だけをソートすることで検索時間の短縮ができることがわかる．

表 2 クラスタリング次元数（8 次元射影）
Table 2 Clustering Dimension (8 projection dimension)

クラスタリング次元	訪問ノード数	距離計算回数	検索時間 (ms)
8	1986	**795512**	58.57
7	1947	795841	58.20
6	**1913**	795913	**58.09**
5	1946	799001	59.67
4	1913	797930	59.94
3	2003	805016	63.13
2	2270	822543	69.85

表 3 クラスタリング次元数（12 次元射影）
Table 3 Clustering Dimension (12 projection dimension)

クラスタリング次元	訪問ノード数	距離計算回数	検索時間 (ms)
12	2196	618481	57.91
11	2215	619031	58.26
10	2198	619090	58.11
9	2198	617993	58.05
8	2194	**617549**	58.14
7	2156	617900	57.97
6	**2122**	618281	**57.75**
5	2158	620432	59.30
4	2128	620673	59.68
3	2237	627728	63.38
2	2549	644652	70.60

表 4　クラスタリング次元数（16 次元射影）
Table 4　Clustering Dimension (16 projection dimension)

クラスタリング次元	訪問ノード数	距離計算回数	検索時間 (ms)
16	2227	511958	59.08
15	2224	512247	59.01
14	2213	512354	58.94
13	2196	512113	58.48
12	2187	512008	58.41
11	2206	512507	58.77
10	2189	512482	58.68
9	2190	511678	58.64
8	2185	**511112**	58.57
7	2151	511477	58.37
6	**2117**	512000	**58.34**
5	2154	514097	59.82
4	2125	514465	60.23
3	2235	521323	63.93
2	2548	537822	71.49

表 5　クラスタリング次元数（20 次元射影）
Table 5　Clustering Dimension (20 projection dimension)

クラスタリング次元	訪問ノード数	距離計算回数	検索時間 (ms)
20	2856	436338	57.31
18	2898	436082	58.15
17	2895	435786	58.30
16	2877	435178	58.00
15	2876	435860	58.16
14	2862	435312	57.82
13	2839	435177	57.56
12	2826	435006	57.35
10	2827	435203	57.48
9	2829	435179	57.48
8	2817	**434295**	57.41
7	2777	435473	57.22
6	**2744**	435355	**57.19**
5	2803	437591	58.86
4	2778	438507	59.46
2	3383	460385	71.06

表 6　クラスタリング次元数（32 次元射影）
Table 6　Clustering Dimension (32 projection dimension)

クラスタリング次元	訪問ノード数	距離計算回数	検索時間 (ms)
32	2766	294782	61.77
27	2771	293776	61.85
26	2747	293538	61.72
25	2767	293981	62.02
20	2837	293771	61.61
16	2861	293055	62.32
14	2820	293247	61.84
13	2821	292564	61.80
12	2790	292816	**61.26**
11	2817	293174	61.60
10	2811	293128	61.65
8	2808	292789	61.49
7	2758	**292226**	61.33
6	**2734**	293653	61.28
5	2804	294802	63.17
4	2767	295499	63.64

4.2　実験 2（クラスタリングと MBR の次元）

クラスタリング次元とともに MBR 次元を射影次元数より減少させて実験を行った．（1）〜（5）に関してそれぞれの実験結果を表 7，表 8，表 9，表 10，表 11 に示す．

結果として，検索時間に関しては（1）〜（5）いずれの場合においても，従来手法と実験 1 に比べて，MBR 次元数も減らしたほうが高速に検索できることがわかる．

表 7　クラスタリングと MBR の次元数（8 次元射影）
Table 7　Clustering and MBR Dimension (8 projection dimension)

クラスタリング次元	訪問ノード数	距離計算回数	検索時間 (ms)
8	1986	**795512**	58.52
7	1949	795875	58.17
6	**1915**	795996	**58.02**
5	1949	799037	59.65
4	1917	798024	59.90
3	2007	805053	63.08
2	2272	822711	68.38

表 8　クラスタリングと MBR の次元数（12 次元射影）
Table 8　Clustering and MBR Dimension (12 projection dimension)

クラスタリング次元	訪問ノード数	距離計算回数	検索時間 (ms)
12	2196	618481	57.87
11	2218	619036	58.19
10	2203	619210	58.02
9	2206	618122	57.93
8	2205	**617633**	58.07
7	2171	618029	57.64
6	**2135**	618494	**57.50**
5	2170	620644	59.16
4	2138	620761	59.63
3	2246	627760	63.22
2	2551	644792	68.38

表 9　クラスタリングと MBR の次元数（16 次元射影）
Table 9　Clustering and MBR Dimension (16 projection dimension)

クラスタリング次元	訪問ノード数	距離計算回数	検索時間 (ms)
16	2227	511958	58.92
15	2226	512331	58.78
14	2216	512453	58.90
13	2203	512260	58.45
12	2196	512169	58.35
11	2218	512719	58.70
10	2203	512851	58.51
9	2206	511928	58.49
8	2205	**511371**	58.57
7	2171	511651	58.36
6	**2135**	512304	**58.20**
5	2170	514439	59.68
4	2138	514585	60.01
3	2246	521411	63.63
2	2551	537848	69.07

表 10　クラスタリングと MBR の次元数（20 次元射影）
Table 10　Clustering and MBR Dimension (20 projection dimension)

クラスタリング次元	訪問ノード数	距離計算回数	検索時間 (ms)
20	2856	436338	57.30
18	2904	436160	57.92
17	2904	435885	57.78
16	2887	435313	57.61
15	2889	435981	57.62
14	2875	435545	57.43
13	2858	435565	57.08
12	2848	435335	56.95
10	2855	435637	56.92
9	2861	435424	57.11
8	2854	**434627**	56.80
7	2814	435743	56.69
6	**2776**	435576	**56.64**
5	2830	437928	58.39
4	2801	438775	59.17
2	3389	460715	70.92

表 11　クラスタリングと MBR の次元数（32 次元射影）
Table 11　Clustering and MBR Dimension (32 projection dimension)

クラスタリング次元	訪問ノード数	距離計算回数	検索時間 (ms)
32	2766	294782	61.77
27	2783	293818	61.42
26	**2759**	293691	61.48
25	2780	294151	62.00
20	2860	294147	61.34
16	2895	293283	61.72
14	2861	293647	61.12
13	2869	292965	61.11
12	2842	293404	60.86
11	2875	293751	61.06
10	2872	293647	60.93
8	2878	293069	60.85
7	2823	**292571**	**60.70**
6	2789	294167	60.74
5	2847	295434	62.48
4	2801	295720	62.88

5. まとめと今後の課題

どの射影次元数の場合でも，本研究で提案するクラスタリング次元を行うことで，ほんのわずかだが，検索時間が短縮できることが確認できた．また，クラスタリング次元だけでなく，MBR 次元数も減らすことで，より高速に検索できる．つまり，クラスタリングと MBR の次元数を射影次元数より小さくしたほうが検索が高速となることがわかる．実験結果より，画像データに対して，実験 2 の 20 次元射影のとき射影データの最初 6 次元だけをソートし，MBR を 6 次元にすることで検索時間が最も速いとなり約 1.2 %の高速化を実現できた．

また，実験 2 が実験 1 と比べて検索時間が速くなる理由は，MBR 次元数が低いほど，検索の空振りが少なくなることが考えられる．検索の空振りとは，MBR と検索範囲が交差した際に，その MBR の中に検索範囲内となるオブジェクトが存在しないことによって無駄なノード訪問をしてしまう現象のことである．

今後の課題として，提案手法に関して音楽データなどの他のデータでも検証を行うことである．その他，ヒルベルトソートする際に，射影データの先頭の次元から一部を取り出すのではなく，任意の次元だけでソートを行う場合の検証や考察が挙げられる．

参考文献

[1] R. Bayer, E. McCreight: Organization and maintenance of large ordered indexes, Acta Informatica Vol. 1, pp. 173–189, (1972).
[2] A.Guttman: R-trees: A dynamic index structure for spatial searching, Proc. ACM SIGMOD, International Conference on Management of Data, pp. 47–57, (1984).
[3] P. Ciaccia, M. Patella, P. Zezula: M-tree: An efficient access method for similarity search in metric spaces, Proc. 23rd Int. Conf. on Very Large Data Bases, pp. 426–435, (1997).
[4] Nick Roussopoulos Timos K. Sellis and Christos Faloutsos. The R+-Tree: A Dynamic Index for Multi-Dimensional Objects. In The VLDB Journal, (1987).
[5] I. Kamel, C. Faloutsos: Hilbert R-tree: An improved R-tree using fractals, In Proceedings of the Twentieth International Conference on Very Large Data Bases, pp. 500–509, (1994).
[6] David Hilbert. Sur une courbe, qui remplit toute une aire plane. Mathematische annalen. (1890).
[7] 田中晶. 空間索引のための多次元データの順序付けの高速化に関する研究. 九州工業大学大学院修士論文， (2001).
[8] T. Shinohara, H. Ishizaka: On dimension reduction mappings for approximate retrieval of multi-dimensional data, Progress in Discovery Science, pp. 224–231, (2002).
[9] 田島圭, 青木隆明, 篠原武. R-tree の検索高速化に対するノードへのデータ配置法に関する研究. 電機関係学会第 62 回九州支部連合大会 (第 62 回連合大会), (2009).
[10] 青木隆明, 田島圭, 篠原武. R-tree の検索高速化に関する研究～ノードへのデータ配置法の提案～. 人工知能学会第 74 回人工知能基本問題研究会 (SIG-FPAI), (2009).