2005－DBS－137（Ⅰ）（8）
2005／7／13

GBD 木のための空間インデックス高速初期構築法の提案

根岸 幸生† 大沢 裕†

†埼玉大学工学部情報システム工学科
〒338-8570 埼玉県さいたま市下大久保 255 埼玉大学工学部情報システム工学科
E-mail: †{negishi,ohsawa}@mm.ics.saitama-u.ac.jp

あらまし 空間データ管理構造 GBD 木における空間インデックスの高速初期構築法を提案する．GBD 木は地理情報システムなどの空間データを管理する上で効率の良いデータ管理構造であるが，空間インデックスの初期構築に長い時間を要する．筆者らはバッチ処理的な手法による空間インデックスの構築法により GBD 木の空間インデックス作成時間を 4 分の 1 に短縮した．また，本手法にて生成された空間インデックスは空間検索時にアクセスするオブジェクト数が従来のものより 40%ほど少ないことを実験により示した．

キーワード GBD 木，空間インデックス，空間データ構造

A Proposal of Fast Initial Spatial Index Construction Method for GBD-Tree

Yukio NEGISHI† Yutaka OHSAWA†

† Department of Information and Computer Sciences, Saitama University
255 Shimo-Okubo, Sakura, Saitama 338-8570, Japan
E-mail: †{negishi,ohsawa}@mm.ics.saitama-u.ac.jp

Abstract The authors propose a new initial construction method for spatial index of GBD-Tree. GBD-Tree is an efficient data structure for management geographical entity. However, GBD-Tree needs long computation time for constructing spatial index at initial construction. 400% improvement in the speed is achieved by using proposal method. In addition, spatial indexes which created by the proposed method are 40% more efficient than previous method in spatial retrieval.

Keyword GBD-Tree，Spatial Index，Spatial Data Structure

1. はじめに

現在，地理情報システムの分野では大縮尺数値地図データの整備がなされつつある．大縮尺数値地図とは 2500 分の 1 や 500 分の 1 といった程度の縮尺の大容量の数値地図であり，家形，道路沿，縁石，側溝といった詳細な地物まで記述されており，数値地図の容量も大きいことが特徴である．一方，筆者らの研究グループでは，時空間地理情報システム(STIMS)[4]を用いて地方自治体の日常的ないくつかの業務で使えるようなシステムのプロトタイプを提案している．これらのシステム上で大縮尺の数値地図を用いたいという要望も大きい．しかしながら，現在の STIMS 上で大容量の数値地図を用いたとき，STIMS の空間データ管理構造である GBD 木のインデックスの初期構築にかなりの時間がかかるという問題がある．

筆者らは，これらの問題点を解決するために，バッチ処理的な手法による空間インデックスの初期構築方を提案する．また，空間インデックスの作成に要した時間と，この手法によって作成された空間インデックスの性能を実験により評価した．

2. GBD 木

GBD 木[1]は R 木[2]や k-d 木[3]と同様に階層的にデータの存在する空間の分割を行い，その分割過程を木構造で管理する図形データ管理構造である．図 1 は GBD 木の構造を示している．GBD 木の木構造は N 個のスロットを持つノードで構成されている．各スロットには子ノード(葉ノードでは図形データ自身)へのポインタと,その子ノードが対応する空間を表す領域式,および子ノードを根とする部分木の全データを含む外接長方形(MBR)を持つ．領域式は図形の空間上での位置と大きさを表現するものである．領域式については 2.2 で詳しく述べる．

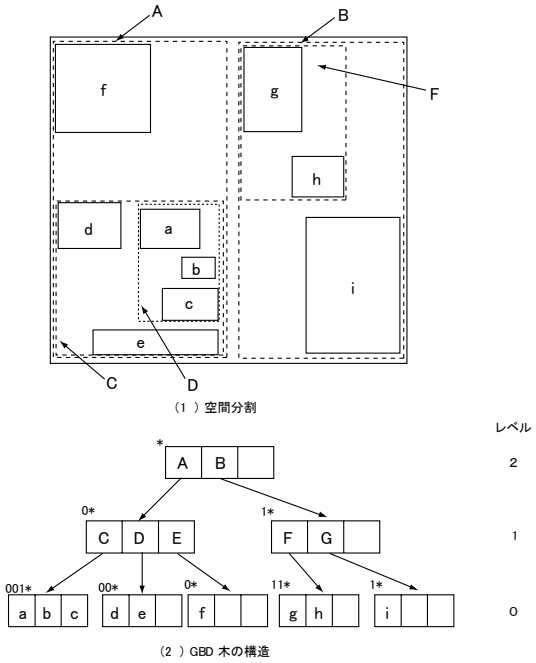

図 1 GBD 木の構造

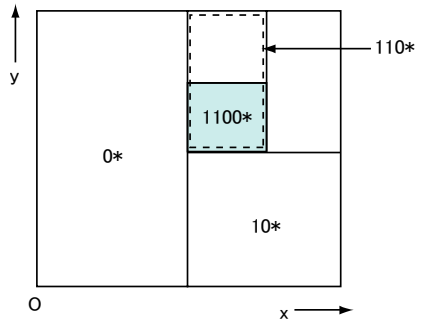

図 2 領域の分割例

2.1. 空間の分割と領域式

GBD 木で管理する領域の形状は分割座標軸を巡回的に変えながら，面積を 2 等分にするように分割を繰り返すことにより得られる正方形,または縦横比が 2:1 の長方形である．この領域の位置と大きさを表すものが領域式である．領域式は「0」「1」およびパターンの終了を表す記号「*」で表現される．図 2 は x 軸，y 軸交互に 2 等分割することにより作られる領域の例を示しており，各領域に対応する領域式を示している．空間分割過程と領域式の対応は次のようになっている．

(1) 領域式中のビット「0」は空間を 2 等分割したときの座標原点側に近い側の領域に，「1」は遠い側の領域に対応している．図 2 では，y 軸と平行な軸で 2 等分割されたとき，領域式中のビット「0」は左半分，「1」は右半分になり，x 軸と平行な軸で 2 等分されたとき「0」は下半分，「1」は上半分に対応している．

(2) 領域式の最も左側のビットが最初の空間分割に対応しており，右側に行くに従い小さな領域の分割に対応している．

また，領域式の大小関係は次のように定義される．

領域式の長さが n1, n2 (n1 > n2)の領域式 R1,R2 において

(1) R1 の上位 n2 ビットが R2 に一致するとき，R1 < R2 とする．

(2) 上位 n2 ビット目まで R1 と R2 のパターンに不一致があるとき，一致しない最初のビットが「0」の側が小さいものとする．

たとえば R1 が「0110011*」，R2 が「0110*」であった場合，(1)より R1<R2 である，また R1 が「001001*」，R2 が「00101*」であった場合，両者の 5 ビット目が異なり，かつ R1 の 5 ビット目が「0」であることから(2)より R1 < R2 である．

2.2. ノード中の領域式と木の探査

GBD 木へデータの投入，または削除をする際に木を根から葉に向けてたどるために，各ノードにおかれている領域式が参照される．GBD 木ではノード中のスロットの配列法に次の条件を設けている．

(1) まず，レベル 2 の各スロットの領域式と Re との包含関係の検索を左から順に行い，最初の Re を包含する領域が見つかったとき，そのスロットが指し示すノードをたどる．Re は「0*」に包含されておりノード A がたどられる．

(2) レベル 1 のノード A には 3 つのスロットが使われている．まず領域式「001*」との比較を行うが Re はこの領域に包含されない，次に「00*」との包含関係が調べられ，Re はその領域に包含されることからノード D がたどられる．

(3) レベル 2 のノード D には 2 つの使用スロットがある．左側から領域式の比較をし，領域式の一致する図形データ d を探し出す．

3. バッチ処理によるインデックス作成方式

従来，GBD 木にデータを追加する場合には，図形データを一つずつ投入してインデックスを作成する手法を用いてきた．しかし，この手法では昨今の大容量データの空間インデックスを作成する場合には長い処理時間を必要とする．このためすべてのデータをあらかじめ領域式を用いてソートしておき，バッチ処理的に空間インデックスを生成する方法を提案する．この，バッチ処理の概要を図 3 に示す．

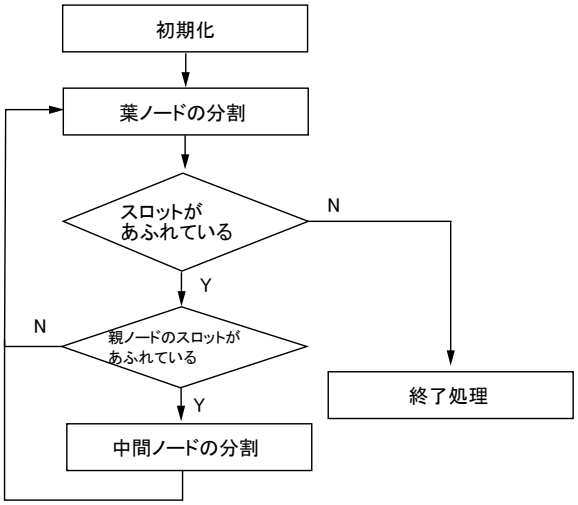

図 3 分割処理の概要

基本的な処理の流れとしては，すべての図形オブジェクトを投入後，再帰的に分割を繰り返し，スロット数が規定のスロット最大数を下回るまで続ける処理である。分割により親ノードのスロットがあふれた場合には親ノードを分割する。具体的な詳細な処理は次のようなものである。

3.1. 初期化

すべての図形オブジェクトを投入し，バッチ処理のための初期化処理をする。具体的な処理は次のとおりである。

(1) GBD 木に挿入するすべての図形の中心点座標の領域式を求める.

(2) ルートノードの下に葉ノード(L)を作成し，ノード L のスロットに，すべての図形を挿入する.

(3) ノード L のスロットを領域式の値で昇べきの順にソートする．(図 5(a))

3.2. 葉ノードの分割

葉ノード（L）の分割は以下の手順で行う．図 4 は図 5 での分割の最初の葉ノードの分割領域式を決めるための過程を図で示したものである.

(1)ノード L のスロットにあるすべての図形の領域式を参照し，ノード L のすべての図形の領域式を含む共通領域式を求める(図 4(a)). この領域式の求め方は 3.4 にて述べる.

(2)求めたノード L の共通領域式の末尾に 1 を加えたものを分割のための領域式（DD)とし，この領域式でノード L を仮に分割する．この仮分割は以下の処理をする.

I. ノード L のスロット一つ一つに対し，分割のための領域式(DD)に各スロットの図形の領域式が含まれているか確認し，含まれているものとそうでないものとの 2 つのグループに分ける(図 4(b)).

II. 2 つのグループに分けたうちの数の少ない側に着目し，数の少ない側の個数が最大スロット数を上回っているならば，ノード L はまだ十分大きいノードであるとみなし，この 2 つのグループで分割を確定する.

III. 2 つのグループに分けたうちの数の少ない側が，最大スロット数を下回っていた場合は，数の大きい側のグループに着目する.

Case A: 数の多い側のグループが，ノード L のスロット数の 3 分の 2 より大きい場合は，数の多い側のグループのみを用いて(1)の処理へ戻り再度分割のための領域式を求める(図 4(b)).

Case B: 数の多い側のグループが，ノード L のスロット数の 3 分の 2 より小さい場合(ただし，ノード L のスロット数より最大スロット数のほうが大きい場合は，最大スロット数の 3 分の 2 より小さいかを判定する)は(図 4(c)), 数の多い側のグループの領域式をすべて含む共通領域式を求め，これを分割のための領域式とし，この領域式に含まれるものとそうでないものでノード L を分割する(図 4(d)).

葉ノードの分割では，数の少ない側がスロット最大数（M)以下になった場合は，数の多い側が，分割対象のノードのスロット数(N)(但し M < N)の 3 分の 2 以下になるまで分割を試行するため，分割結果 R は 1/3(N) < R < 2/3(N) を満たす．分割処理が進むにつれノードのスロット数は最大スロット数に近づくため(M ≒ N)となり，もっとも悪いケースとなる M+1 個のオブジェクトの分割の場合でも，1/3(M+1) < R < 2/3(M+1) を満たす.

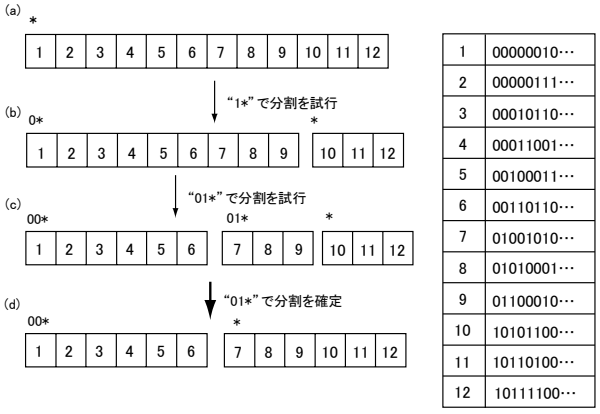

図 4 葉ノード分割の例

3.3. 中間ノードの分割

非葉ノードですべてのスロットが使われているとき，その子ノードで分割が発生すると，その非葉ノー

ドはスロットの最大数を超えて子ノードを持つことになり，あふれが生じる．この場合に，次に示す方法で中間ノードの分割を行う．

(1) 分割の対象となる非葉ノード(T)の各スロットの領域式を参照しつつ．その領域式が自分を含めいくつのスロットを内包しているか調べる．

(2) 最も(N+1)/2 に近い数を内包する領域式を新しい領域 A の領域式とし，領域 A に内包される子ノードと，それ以外の子ノードに分割する．

3.4. ノードの共通領域式の取得

ノード中のスロットのすべての領域式を含む領域式である共通領域式は次のように求める．

(1) 求めるノードの 1 番目のスロットの領域式を取得する．これを UD とする．

(2) 次のスロットの領域式を取得し，この領域式と UD の最も左の桁から順に調べる．

Case A: 桁の数字が同じであった場合は，ひとつ右側の桁に移り，その桁を判定する．

Case B: 桁の数字が異なっていた場合は，その桁の手前までを共通の領域式とし，この領域式の値で UD を更新する．

(3) 2 の処理を，n 番目のスロットまで繰り返す．

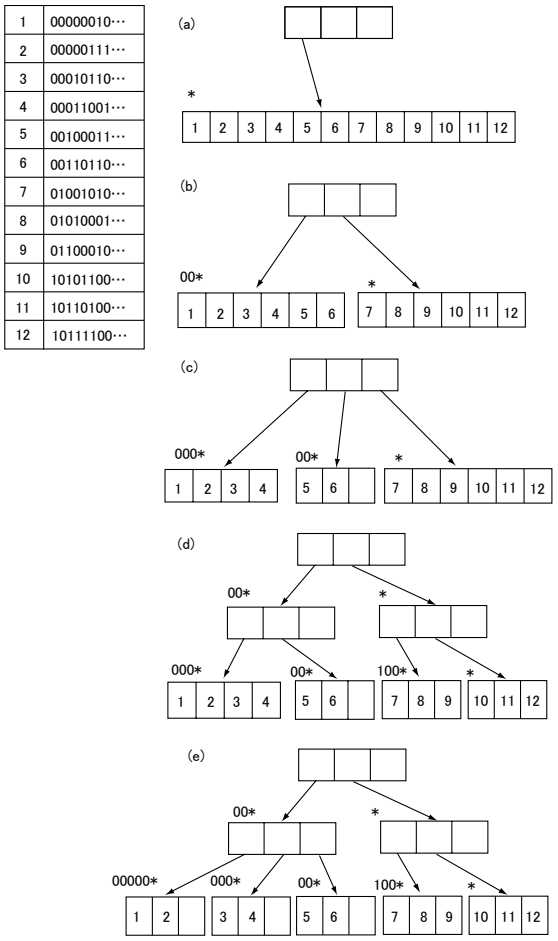

図 5 バッチ処理による分割の例

4. 性能の評価

4.1. 各手法による性能評価

前章にて述べた手法を実装し実験により評価した．実験内容は当研究グループで開発している，時空間情報管理システム:STIMS[4]の空間データファイルを読み込み，空間インデックスファイルの作成にかかった時間を測定した．実験には，800KB から 120MB までの 4 種類の数値地図を用いた．数値地図名，データサイズ，オブジェクト数を表 1 に示す．計算機へのインプリメントでは領域式を 64 ビットとし，領域式の長さを 8 ビットで表現した．実験には，今回提案した手法と従来の STIMS で採用されている手法[1]の 2 種類について，それぞれ空間インデックス作成にかかった時間を測定した．測定に用いた実験環境を表 2 に示す．

実験結果を表 3 に示す．また，本実験では，GBD 木の最大スロット数を 50 とした．このスロット数は，現在 STIMS で用いてられている最大スロット数である．

提案手法は，どの数値地図においても良好な結果が得られていることがわかる．また，従来手法との比較では，どの数値地図においても約 4 倍程度，速度が向上した．

表 1 実験に用いた数値地図

数値地図	縮尺	データサイズ	オブジェクト数
JMCマップ KS5339	1/200,000	840KB	8,583
数値地図 25,000 さいたま市	1/25,000	1.4MB	27,074
トロピカルテクノセンター 沖縄市	1/2,500	10MB	151,699
J-Mapple 200,000 日本全国	1/200,000	120MB	610,756

表 2 実験に用いた計算機のスペック

CPU	Pentium 4 2.8CG
RAM	2 GB
OS	Windows XP Professional

表 3 空間インデックス作成に要した時間

ベースマップ	従来手法	提案手法
JMC KS5339	2.3	0.6
JSGI さいたま市	7.0	1.8
TTC 沖縄市	39.4	10.4
J-Mapple 200000	164.4	44.3

単位[秒]

4.2. 最大スロット数を変化させた場合の性能評価

次に，ノードあたりの最大スロット数を変化させた場合についての評価実験を行った．実験内容は，先の実験と同様に STIMS のデータファイルを読み込み，空間インデックスファイルの作成にかかった時間を測定した．実験に用いた数値地図は先の実験で用いたトロピカルテクノセンターの那覇市のデータを用いた．測

定は今回提案した手法と，従来の手法[1]の２つについて，最大スロット数を 25 から 1000 まで変化させインデックスの構築に要した時間を測定した．実験環境は先の実験環境と同じである．実験結果を図 6 に示す．

提案手法は，スロット数が多くなるに従い処理時間が緩やかに減少している．一方，従来の手法ではスロット数が多くなるに従い処理時間が増加することがわかる．この結果は，提案手法ではすでにすべてのスロットを領域式でソートし，その後ノード分割をする．このため，スロット数が増えるに従いノード分割数が減り，処理時間が減少するためである．

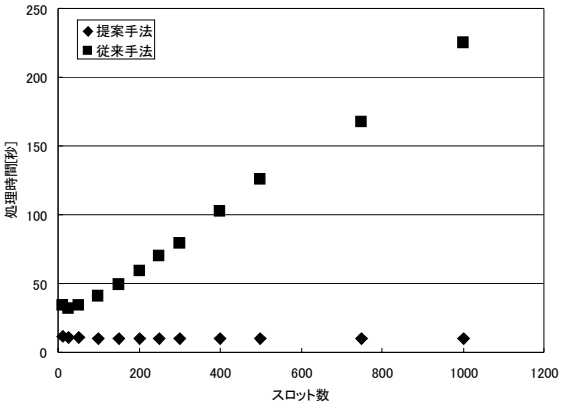

図 6 バッチ処理による分割の例

一方，従来の手法ではデータを投入するごとに，木をたどり,適切な位置のノードを下る必要があるため，スロット数の大きいノードではノードあたりの判定処理に時間がかかり，処理時間が増加する．

実験結果より，提案手法は従来手法より高速に GBD 木のインデックスを作成することができる．特に GBD 木のノードの最大スロット数が多い場合には，提案手法がより高速にインデックスを作成することができる．

4.3. 検索性能の評価

従来手法である逐次的なインデックス作成方式と，提案手法であるバッチ処理的なインデックス作成方式の両者のインデックスの性能を比較するため 3 つの実験をした．まず，それぞれの方式によって作成されたインデックスのスロットの占有率の比較をした．加えて，空間データの検索時にたどったノード数と，アクセスしたオブジェクト数を比較する実験をあわせて行った．さらに，本手法に適したスロット数を求めるため，スロット数を変更した場合のインデックスサイズの変化についても実験をした．

4.3.1. スロット占有率の比較

従来手法との性能を比較するため，スロットの占有率を比較した．スロット占有率とは，ノードのスロットのうち子ノードや空間データに割り当てられている有効なスロットの占める割合である．この実験では，逐次投入方式とバッチ処理のそれぞれの方式で作成した空間インデックスをたどり，各ノードにおいて有効なスロットを調べ，スロット占有率を求めた．空間データにはトロピカルテクノセンターの那覇市のデータを用いた．また，スロット数は 50 とした．

従来手法ではスロット占有率の平均値が 60.2%であったのに対し，提案手法ではスロットの占有率が 67.2%であり，スロット占有率が高いことがわかる．これは，逐次投入方式ではノード分割をした後にも空間データが挿入されることで，最適な空間の分割ができず,分割後のノードに偏りができてしまうのに対し，バッチ処理ではあらかじめすべての空間データが投入されていることにより，バランスよく空間を分割することができるためである．

4.3.2. 検索効率の比較

提案手法によって作成したインデックスの検索効率を調べるため，評価実験を行った．実験内容は，数値地図中にあらかじめ無作為に 100 箇所の地点を設定し，その範囲を中心とした 10,000×10,000 の正方形領域の検索処理をしたときの，たどったノード数とアクセスしたオブジェクトの数を調べ，その平均値を求めた．この正方形領域は数値地図の全空間サイズの約 0.1%にあたる，数値地図には先の実験と同じ那覇市のデータを用いた．数値地図の幅は 10,200,000，高さは 8,000,000 である．また，スロット数は 50 で実験をした．実験結果を表 4 に示す．

表 4　空間インデックスへのアクセス数

	従来手法	提案手法
オブジェクトアクセス数	5654	3804
葉ノードアクセス数	184	109
中間ノードアクセス数	108	78

従来手法の逐次投入に比べ，提案手法ではたどるノード数，アクセスするオブジェクト数ともに平均値で 40%ほど少ない．この理由は次の２点である．

(1)　提案手法ではスロット占有率が高く，より効率的な検索ができるため，検索時にたどるノード数が少なくて済む．

(2)　提案手法では，すべての空間データを投入した後に空間分割をしていることから，無駄のない空間分割ができているので，たどるノード数が少なくて済む．

4.3.3. スロット数とインデックスサイズの比較

提案手法で作成されるインデックスに適したスロット数を調べるため，スロット数を変えながら GBD 木のインデックスを作成し，そのインデックスサイズを比較した.実験内容は,今回提案した手法を用いて，最大スロット数を 25 から 2000 まで変えてインデック

スを作成し，GBD 木のインデックスのサイズを測定した．数値地図には先の２つの実験と同じ那覇市のデータを用いた．実験結果を図 7 に示す．

実験結果より，最大スロット数が増加すると，インデックスサイズは増加減少を繰り返しながら，緩やかに減少する結果が得られた．この現象は次のように説明できる.今回提案したバッチ処理的な分割手法では，最も理想的なデータが与えられた場合は，ノードは常に二等分に分割されていく．このような理想的な状況では，最大スロット数(N)を

N=(全空間オブジェクト数/ 2^k) (k = 1,2,3…)

にすると，スロット占有率は 100%になる．実際の分割では上記のような理想的な分割は起こりえないが，分割は平均的には半分程度に分割されていると考えられるため，最大スロット数を上記の値にすることで，高いスロット占有率が得られ，インデックスサイズが減少する結果が得られる．

5. まとめ

本稿では，GBD 木の空間インデックスの初期構築を高速化するための手法について述べた．GBD 木の各ノードは，データ投入や削除のための補助情報である領域式と，データ検索のための子ノードをすべて包含する MBR を持つ．この２つの補助情報により，検索性能を犠牲にすることなく，データ更新を高速に行うことができる．筆者らは，GBD 木の空間インデックスを高速化するため，すべてのオブジェクトを領域式順にソートし，領域式の共通部分を求め，再帰的に分割を繰り返すことによるバッチ処理的な手法で GBD 木の空間インデックスを作成する手法を提案した．また，本手法を実際の計算機上にインプリメントし，時空間情報管理システム(STIMS)の空間インデックス作成に要する時間を測定した．

提案手法は従来の手法より 4 倍程度高速にインデックスを構築することが可能である．またスロット数が増えた場合にはさらに高速に構築できる．このため，大規模データや精度の高い大縮尺の地図データの空間インデックスを作成する場合には本手法が有用である．

また，あわせて両手法の検索性能の評価を行った．まず，両手法のスロット占有率の比較をした結果，提案手法は従来手法より高いスロット占有率が得られることを示した．また，実際の検索処理においても，提案手法は，スロット占有率の高さから，たどるノード数，アクセスするオブジェクト数ともに従来手法より少なくなる結果が得られた．また，本手法に適した最大スロットのサイズを調査するため，1 ノードあたりの最大スロット数を変更してインデックスサイズの変化を測定した結果，本手法を用いた場合，最大スロット数(N)を

N=(全空間オブジェクト数/ 2^k) (k = 1,2,3 …)

にすると高いスロット占有率が得られることを示せた．

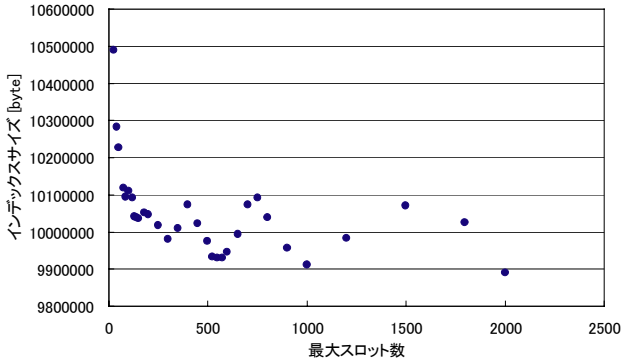

図 7 最大スロット数と
空間インデックスサイズの関係

文　献

[1] 大沢裕，坂内正夫: 2 種類の補助情報により検索と管理性能の向上を図った多次元データ構造の提案，

[2] Bentley J.L.: "Multi Dimesional Binary Search Trees Used for Associative Searching", Commun. ACM, 18, 9 (1975).

[3] Antonin Guttman : "A Dynamic Index Structure for Spatial Searching", Proc. of the 1984 ACM-SIGMOD International Conf., pp. 47-57 (1984)

[4] 大沢裕，長島敦: トポロジー暗示型時空間情報管理システム：STIMS，第１２回機能図形情報システムシンポジウム講演論文集, pp. 27-36 (2000).

[5] 根岸幸生，青木秀晃，笠原直，郭薇，川崎洋，大沢裕: 時空間管理のための地理情報システム STIMS，電子情報通信学会技術研究報告，Vol.~103 No.191 pp. 7-12 (2003).

[6] Yutaka Ohsawa, Yukio Negishi, Hideaki Aoki, Guo Wei, Hiroshi Kawasaki: A Geographic Information System for Spatio-temploral Data Management: STIMS, ASGIS03, pp. 73-80 (2003).

[7] 畑山満則，松野文俊，角本繁，亀田弘行: 時空間地理情報システム DiMSIS の開発, GIS 理論と応用, 7(2), pp. 25-33 (1999).

[8] Burrough,P.A, McDonnell,R.A.: Principles of Geographical Information Systems, NewYork, Oxford University Press (1998).

[9] 大伴真吾，大沢裕: 位相構造を持たない地理情報システムに関する考察,第 8 回機能図形システムシンポジウム, pp. 55-62(1997).

[10] 根岸幸生，川崎 洋，大沢 裕: 空間インデックス GBD 木の高速初期構築法，FIT2004，第二分冊 pp.35-36 (2004.9)

[11] Yukio Negishi, Yutaka Ohsawa: A Proposal of GBD-Tree Fast Initial Construction for Spatial Index, ASGIS 2005