社団法人 情報処理学会　研究報告
IPSJ SIG Technical Report

2004－DBS－134　(I)　(8)
2004／7／13

拡張可能グリッドファイルにおける最近傍検索の改善

三　好　涼　介[†]　三　浦　孝　夫[†]　塩　谷　　勇[††]

本稿では、拡張可能グリッドファイルにおける最近傍検索の性能改善を行う手法を提案する。具体的には論理領域単位で空セルを回避し、また各領域ごとの MBR 情報を付与することにより、ページアクセス数を低減できることを示す。

Querying Nearest Neighbor Data on Extensible Grid Files

Ryosuke MIYOSHI,[†] Takao MIURA[†] and Isamu SHIOYA[††]

Nowadays we see a variety of spatial applications. However, as well-known, it is hard to realize both efficient and dynamic access techniques yet independent of data distribution. In this investigation we propose a sophisticated access mechanism based on grid file structure extended by avoiding null region and gibing MBR to each region, and we show several suitable aspects as the spatial structure empirically.

1. 前　書　き

近年のデータベースの分野において、地理データや CAD データなどの空間データを処理するデータベースシステムの開発が盛んに行われており、現在多くの空間データ処理システムが提案されている[3),10)]。現在の空間データ処理システムは SS 木や SR 木などの R 木を基盤とした階層的な空間索引構造が最も主流である。

R 木[4)] は、空間データ集合を格納すべき最小包囲長方形 (Minimum Bounding Rectangle, MBR) と呼ばれる超長方形を領域分割方式に従って再帰的、階層的に分割する多分木である。従来の階層索引構造と違い MBR の重なりを許容するため、平衡した木となることが知られている。しかし、重なりを許容することによって、質問によってはトラバースが発生したり、動的挿入、更新時の処理負荷が増加するなどの問題がある。

SR 木[6)] は最小包囲球 (Minimum Bounding Sphere, MBS) と MBR を組み合わせ、重なる部分をノードとしたものである。径も体積も小さくなり検索に優れたものとなっている。しかし R 木と同じようにノードの重なりを許容するため、平衡木にはなるがトラバースの発生やデータ更新の点などの問題がある。

本稿では拡張可能グリッドファイルを提案する。グリッドファイルは、ファイルの負荷率を考慮して空間を排他的に分割していく非階層的索引構造である。分割された空間はグリッドによって管理され、すでに報告したように完全一致検索や挿入操作は最も高速に行える手法である[7)]。反面、空バケットの存在を許すため最近傍検索のアクセスコストは高くなるという問題がある。本論文では、最近傍検索の性能を改善するため拡張可能グリッドファイルを導入する。具体的にはバケットの索引に、そのバケットのデータ内在判定と MBR 情報を付与し、この構造に順応する検索手法を提案する。これにより検索における無駄なページアクセス数を低減できることを示す。

2,3 章では拡張可能グリッドファイルの構造とその処理の流れを述べる。4 章で拡張可能グリッドファイルにおける検索の手法について論ずる。5 章では実験結果を示し、6 章では関連研究を要約する。

2. 拡張可能グリッドファイル

2.1 グリッドファイルの問題

グリッドファイル[8)] は、各ファイルの管理領域をデータ分布に従って変動させることで任意のファイルへの

† 法政大学 工学研究科 電気工学専攻　〒 184–8584 東京都小金井市梶野町 3-7-2
Dept.of Elect.& Elect. Engr., HOSEI University　3-7-2, KajinoCho, Koganei, Tokyo, 184–8584 Japan
†† 産能大学 経営情報学部　〒 259–1197 神奈川県伊勢原市上粕屋 1573
Department of Management and Information Science, SANNO University　1573, Kamikasuya, Isehara city, Kanagawa 259–1197 Japan

みのデータの偏りを防ぐことができる空間索引構造である。グリッドファイルでは、ファイルの索引を見出すために分割される空間をグリッド空間と呼び、その分割され索引となる領域をグリッドと呼ぶ。またデータポインタを挿入するファイルをバケットと呼ぶ。図1では2次元空間が排他的に6個のグリッドに分割され、それぞれがバケットで管理されている状況を示している。

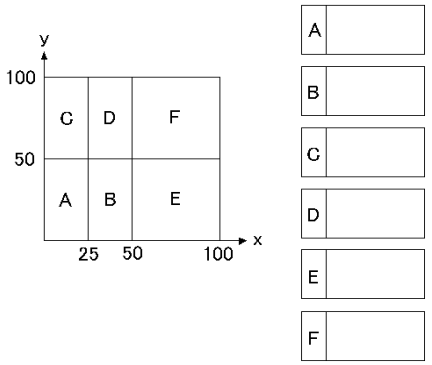

図1 グリッドファイル

グリッドファイル構造は排他的、非階層的な索引構造であるため、利点はキーに対応するグリッドが得られれば各種操作は効率良く行われ、完全一致検索や挿入操作では1回のページアクセスで操作を完了することができる。反面、グリッド空間の分割が進むにつれて、キーに対応するグリッドを検索するためにかかる参照回数が非常に大きくなる。

また検索、特に最近傍検索については次の2つの問題がある。1つは、グリッドファイルが空バケットの存在を許す構造であるため、空バケットが検索範囲内にあると無駄なバケットアクセスが発生してしまう点である。特に最近傍検索で検索点が空バケットの領域範囲内で、周囲の索引も同じバケットを指している場合データの内在判定のために何回もこの空バケットにアクセスし、結果コストの高いものとなってしまう。第2の問題は、すべての領域はいずれかのバケットで管理されており、バケット内のデータにどんな偏りがあろうとも最低1回は該当バケットをアクセスしてしまうことにある。例えばあるバケットでは右上にのみデータが偏っているにもかかわらず、左下に対しての検索は無駄なバケットアクセスを増やすだけである。この2つの問題は、実データのような非一様分布となるデータで顕著に見られる。

本稿では、これらの問題が拡張可能グリッドファイル構造を用いて解決できることを示す。実際、まずグリッド空間の分割に伴う参照回数の急激な上昇を回避するため、拡張ハッシュ手法[5]を適応させる。さらに、検索に対するアクセスコストを低減させるため、バケットへの索引にそのバケットのデータ内在判定とMBR情報を与える。データ内在判定は前者の検索問題に対するものであり、これにより直接バケットにアクセスを行わずともバケット内のデータの有無を判定することが可能となる。MBR情報は後者の検索問題に対するものである。これにより座標空間での検索に対する処理を行う前に、グリッド空間で検索範囲(点)とそのバケットのMBR情報が重なるかどうかを調べることで無駄なバケットアクセスを低減できる。

2.2 データ表現とデータ構造

拡張可能グリッドファイル (Extended Grid File, EGF) では、n 次元の空間データに対し、まず各次元の値を2進数で表現しそれぞれの上位 P_n 桁を切り出す。切り出された数がバケットと呼ばれるデータを収納するファイルへの索引となる。これは、グリッドファイルにおけるグリッドに相当する。さらにこの個々の索引において、その索引の指すバケットのデータ内在判定と、バケット内に1つだけMBRの存在を許したときのそのMBR情報を付与する。この配列をディレクトリと呼ぶこととし、対応するバケット, データ内在判定,MBRの頂点 $M\_\_L$,M_H(ただし $M_L = (l_1, l_2, ..., l_n)$,$M_H = (h_1, h_2, ..., h_n)$,$l_i \leq h_i$)と表される。また、ディレクトリは $2^{\sum P_n}$ 個の索引からなる。

また、値から切り出される桁数 P_n はバケット数に応じて伸縮する。そして、その切り出された値より索引が指定され、その索引に対応するバケットを取り出し各種操作を行う。挿入操作の場合はディレクトリの更新が実行される可能性があり、データのないバケットにデータが挿入されるとき、若しくはMBRの領域外の座標のデータが挿入されるときに更新される。

データのないバケットにデータが挿入される場合、そのバケットを指す索引すべてに対してデータ内在判定を更新する。その際MBRも挿入された点を特異なMBRとして書き込む。MBRの領域外の座標のデータが挿入される場合、そのバケットを指す索引すべてに対してMBRを更新する。

例1 図2は、x 軸, y 軸共に値の上位1桁をディレクトリへの索引としている。索引はそれぞれ、(0,0) はバケットA、(0,1) はバケットB、(1,0),(1,1) はバケットCへのポインタとなる。また、(0,0) はバケットA

の、(0,1) はバケット B の、(1,0),(1,1) はバケット C のデータ内在判定とそれぞれのバケット内の MBR 情報を持つ。ここで $R = (51, 118) = (0110011, 1110101)$ となるデータ R の挿入を考える。いま $P_x = P_y = 1$ であるから R_x, R_y の上位 1 桁を取り出し索引を得る。$(0, 1)$ はバケット B を指しているのでバケット B を取り出し挿入を行う。もしバケット B が空バケットである場合、$(0, 1)$ のデータ内在判定を真にし、MBR 情報を $((51, 117), (51, 117))$ に更新する。また、バケット B の MBR 情報が $((40, 120), (50, 125))$ である場合、$((40, 117), (51, 125))$ に更新する。□

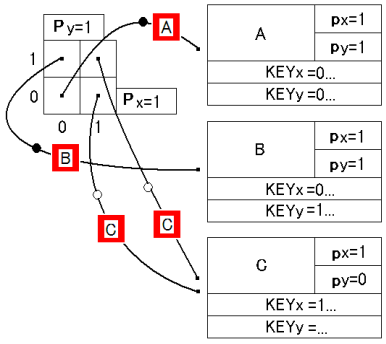

図 2 拡張可能グリッドファイル

3. データ更新による構造変化

一定のバケットへのデータの偏りを防ぐため、バケットにはあらかじめ最大容量 B が決められており一定のデータ数を超えると拡張操作が行われる。このとき複数の索引から参照されているバケットが一杯になるとバケット自体が分割され、単索引から参照されているバケットが一杯になるとディレクトリが拡張される。したがってバケットも自身の管理している領域の桁数 p_n を記憶している。

バケットにおいてデータ挿入によってあふれが生じる場合、$P_x = p_x$ 且つ $P_y = p_y$ となるバケットのあふれ発生時であればディレクトリ拡張操作が、それ以外のバケットのあふれ発生時ならばバケット分割操作が行われる。バケット分割時はあふれの生ずるバケットの管理する領域を分割軸で 2 等分割する。片方を前のバケットが、もう一方を新しく作成されるバケットが管理するとし、挿入データも含めてデータを移動させる。ディレクトリ拡張時は拡張軸の索引を 2 倍に拡張してからバケット分割を行う。データを移動させたあと、前のバケットに管理されていた領域の索引をそれぞれのバケットに合わせて更新する。バケット分割時の分割軸の選択は、$P_n - p_n$ が同じならばデータ分布の大きい軸を、それ以外なら $P_n - p_n$ の値の小さい軸を分割軸とする。ディレクトリ拡張時の拡張、分割軸の選択は P_n の最も小さい値の軸を、最小の P_n の値を持つ軸が 2 つ以上存在するなら第 1 次元に最も近い軸を分割軸とする。

例 2 図 2 のバケット C があふれた場合バケット分割操作が行われ図 3 の状態となり、バケット B があふれた場合ディレクトリ拡張操作が行われ図 4 の状態となる。□

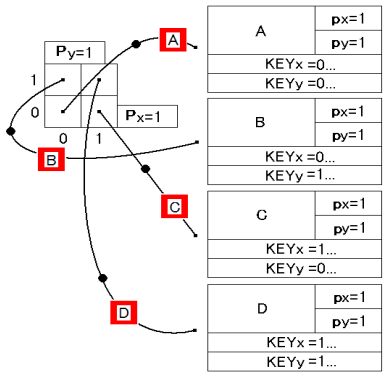

図 3 バケット C 分割後のキー分布

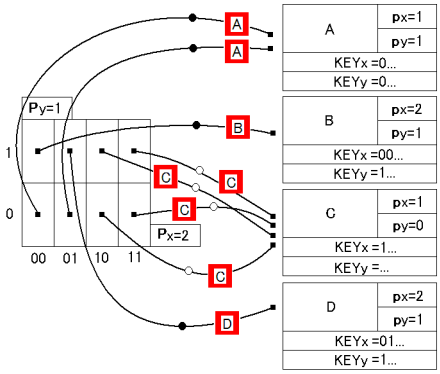

図 4 ディレクトリ拡張後のキー分布

4. EGF 構造の検索手法

4.1 完全一致検索

完全一致検索とは、各次元のキーをすべて指定し、これをすべて満たす空間情報を求める検索をいう。検索点のそれぞれ上位 P_n 桁を取り出し索引を得る。取

り出した索引において判定を行う。索引に対応するバケットの内在判定が真で、且つ検索点が MBR 内にあるなら、バケットにアクセスしバケット内のすべてのデータに対して比較を行う。

4.2 範 囲 検 索

範囲検索とは、ある指定した範囲に含まれるすべてのレコードを求める検索をいう。一般に検索範囲はすべての軸 i について、$LOW \leq a_i \leq HIGH$ の形で指定される。検索方形領域の頂点すべての上位 P_n 桁を取り出し索引とし、グリッド空間に射影させた範囲にある索引について判定を行う。索引に対応するバケットの内在判定が真で、且つ検索範囲が MBR と重なるバケットをアクセスリストに入れ、それ以外はエリミネイトリストに加えておく。すべての索引の判定終了後、アクセスリストにあるバケットのデータに対して実際の判定を行う。

例 3　図 5 で範囲検索を行うとする。灰色で囲まれた検索範囲をグリッド空間に射影させ、$(01,0)$,$(01,1)$,$(10,0)$,$(11,1)$ のディレクトリを得る。$(01,0)$ に対応するバケット A は内在判定が真で、且つ MBR が検索範囲と重なる。アクセスリストにバケット A は含まれていないのでアクセスリストに入れておく。$(01,1)$ に対応するバケット D も内在判定が真で、且つ MBR が検索範囲と重なる。バケット D もアクセスリストに含まれていないので入れておく。$(10,0)$ に対応するバケット C は内在判定は真であるが、MBR が検索範囲と重ならない。よってバケット C はエリミネイトリストに入れておく。$(10,1)$ に対応するバケット C はすでにエリミネイトリストに入っているので判定を行わない。以上よりアクセスリスト内のバケット A,D にアクセスし、実際に検索範囲内にあるデータを算出する。□

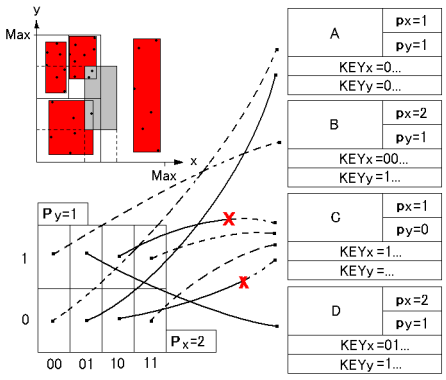

図 5　範囲検索

4.3 最近傍検索

各次元のキーをすべて指定して、それに最も近い位置にあるレコードを求める検索を最近傍検索という。検索点のそれぞれ上位 P_n 桁を取り出し索引とする。取り出した索引において判定を行い、内在判定が真なら索引の対応するバケット内のデータで距離計算を行い仮の最近点 Q を算出する。内在判定が偽なら、指定された索引の周囲の周囲の索引に対応するバケットにアクセスし、データが存在するバケットを検出するまで索引の検索範囲を拡げ、検出されたバケットの中から仮の最近点 Q を算出する。Q 算出後、質問点と Q との距離を L とし、質問点を中心とした辺長 $2L$ の正方形領域に対して範囲検索を行う。この時、Q を算出する際に調べた索引は検索対象から排除する。得られたデータより真の最近点 Q' を算出する。

例 4　図 6 で最近傍検索を行うとする。検索点 P よりそれぞれ上位 P_x, P_y 桁を取り出し索引 $(00,0)$ を得る。$(00,0)$ に対応するバケット A は内在判定が偽であるためエリミネイトリストに入れておく。アクセスリストが空なので索引の検索範囲を拡げ、索引 $(00,1)$,$(01,0)$,$(01,1)$ を得る。$(00,1)$ に対応するバケット B は内在判定が真であるからアクセスリストに入れておく。$(01,0)$ の指すバケット A はすでにエリミネイトリストに入っているので判定を行わない。$(01,1)$ に対応するバケット D は内在判定が真であるからアクセスリストに入れておく。以上よりアクセスリスト内のバケット B,D にアクセスし、仮の最近点 Q を算出する。

次に線分 PQ の距離を L とし、P を中心とした辺長 $2L$ の正方形領域に対して範囲検索を行い、索引 $(00,0)$,$(00,1)$,$(01,0)$,$(01,1)$,$(10,0)$,$(10,1)$ を得る。この時、$(00,0)$,$(00,1)$,$(01,0)$,$(01,1)$ は Q 算出時に判定、アクセスを行っている。よって $(10,0)$,$(10,1)$ にのみ判定を行う。$(10,0)$ に対応するバケットは内在判定が真で、且つ検索範囲と MBR が重なる。よってアクセスリストに入れておく。$(10,1)$ に対応するバケットはすでにアクセスリスト内に含まれているので判定を行わない。以上よりアクセスリスト内のバケット C にアクセスし、真の最近点 Q' を算出する。□

5. 性 能 評 価

本章では EGF の有用性を実験により評価する。

5.1 実 験 環 境

使用する 2 次元の点データ 119,898 点は、各レコードが 14 バイト長 (2 つの 7 バイト long integer 型項

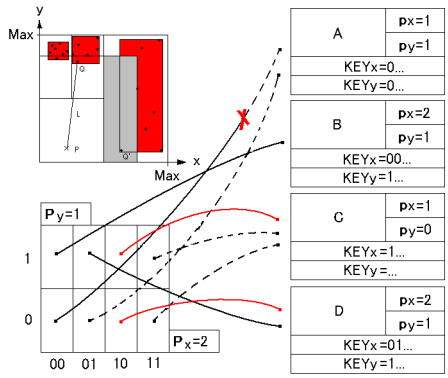

図 6 最近傍検索

目からなる) のニューヨーク、ボストン、フィラデルフィア周辺の郵便アドレス 119,989 件である☆。これを 100000+19898 点、60000+59898 点の二通りに分割し初期生成 (静的挿入)、追加挿入 (動的投入) のそれぞれで性能を調べるとする．また、質問点 10000 点は乱数で、範囲検索時は質問点を中心とした辺長 10000 の方形領域を範囲とする。1 バケットを 1 ページとし、$K = 20$、$B = 4, 8, 16, 32, 64$ Kbyte として実験を行う。

5.2 実　験　1

点データ 100000 点と 60000 点で一括生成を行い各検索を実行する。各検索のページアクセス回数の計測を行い、SR 木との比較を行う。また EGF における MBR の有用性を検証するため、MBR 情報を持たない EGF とも比較を行う。実験結果を表 1 に示し、グラフを図 7 に示す。ここでの比率は SR 木に対する EGF の比をあらわす。

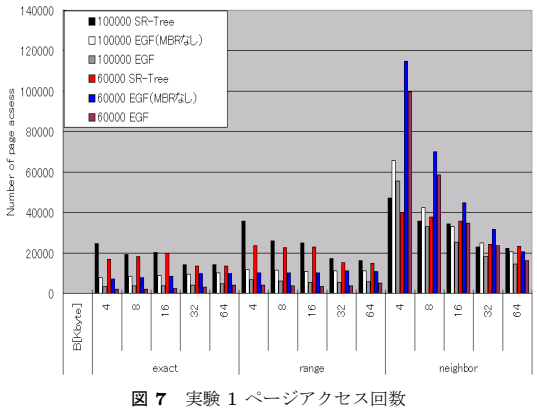

図 7 実験 1 ページアクセス回数

☆ 本実験では www.rtreeportal.org で公開されている地図情報を用いた

com	B	SR 木	EGF(MBR なし)	EGF	比率
100000					
EQ	4	24712	7702	3340	7.4
	8	19274	8441	3557	5.4
	16	20267	8588	3694	5.5
	32	14129	9557	4168	3.4
	64	14267	10000	4808	3.0
RQ	4	35532	11797	6849	5.2
	8	25911	11314	5998	4.3
	16	24740	10682	5432	4.6
	32	17041	11033	5374	3.2
	64	16199	11053	5748	2.8
NNQ	4	47269	65458	55415	0.85
	8	35709	42293	33006	1.1
	16	34395	33005	25085	1.4
	32	22789	24751	18162	1.3
	64	22365	20573	14517	1.5
60000					
EQ	4	16973	7216	1874	9.1
	8	18324	7891	2178	8.4
	16	19819	8468	2309	8.6
	32	13405	9886	2925	4.6
	64	13455	9853	4197	3.2
RQ	4	23553	10027	4164	5.7
	8	22471	10017	3782	6.0
	16	22736	10018	3397	6.7
	32	15243	11054	3760	4.1
	64	14756	11670	4882	3.0
NNQ	4	39914	114702	99746	0.40
	8	37610	69842	58627	0.64
	16	35839	44900	34747	1.0
	32	24295	31675	23535	1.0
	64	23297	20643	16305	1.4

EQ:完全一致検索 RQ:範囲検索
NNQ:最近傍検索

表 1 実験 1 ページアクセス回数

5.3 実　験　2

実験 1 で構築した一括生成ファイルに対し、それぞれ残りの点データ 19898 点と 59898 点を投入後各検索を実行する。各検索のページアクセス回数の計測を行う。ここでも、MBR 情報を持たない EGF に関する実験も行う。実験結果を表 2 に示し、グラフを図 8 に示す。

5.4 考　　察

各実験についての考察を行う。実験 1 では各検索のアクセスコストの低さが確認できた。完全一致検索、範囲検索では SR 木に対してすべての結果で性能向上が見られ、それぞれ最大で $B = 4$ の 60000 点で約 9.1 倍、$B = 16$ の 60000 点で約 6.7 倍のアクセスコストの比率となった。これは本稿で提案する EGF ではデータ内在判定及び MBR 情報により 1 回もアクセスせずに検索を終える場合があり、結果平均で SR 木に勝る。また MBR のない EGF でのページアクセス回数の結果から、特に MBR 情報がアクセスコストの低

com	B	SR 木	EGF(MBR なし)	EGF	比率
19898					
EQ	4	28404	7926	3609	7.9
	8	20838	8508	3817	5.5
	16	22810	8546	3923	5.8
	32	23324	9557	4295	5.4
	64	15380	10000	4911	3.1
RQ	4	42068	12356	7493	5.6
	8	29653	11600	6502	4.6
	16	29022	10769	5826	5.0
	32	27580	11142	5610	4.9
	64	18164	11125	5913	3.0
NNQ	4	53149	65405	55708	1.0
	8	37802	42744	34121	1.1
	16	38463	32765	25861	1.5
	32	37322	24564	18328	2.0
	64	23195	20218	14276	1.6
59898					
EQ	4	28734	8045	3621	7.9
	8	21945	8471	3821	5.7
	16	24124	8570	3917	6.2
	32	26381	9518	4388	6.0
	64	15660	9853	5584	2.8
RQ	4	43319	12460	7502	5.8
	8	31142	11592	6539	4.8
	16	30544	10802	5811	5.3
	32	30722	11084	5673	5.4
	64	18529	10936	6643	2.8
NNQ	4	55001	67033	56967	1.0
	8	38535	41916	34036	1.1
	16	41572	32556	25608	1.6
	32	38716	25814	19027	2.0
	64	23630	17094	14071	1.7

表 2 実験 2 ページアクセス回数

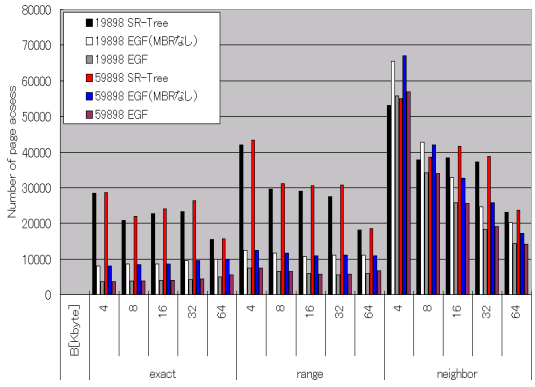

図 8 実験 2 ページアクセス回数

減に寄与している。

実験 2 では、EGF の高い動的特性の保持が確認できた。挿入を行った後でも完全一致型検索、範囲検索において SR 木に対しすべての結果でほぼ同等かそれ以上の性能であり、それぞれ最大で $B = 4$ の 59898 点投入で 7.9 倍、$B = 4$ の 59898 点投入で 5.7 倍のアクセスコストの比率となった。データ内在判定と MBR 情報によるアクセスコストの削減と考えられる。また、実験 1 のページアクセス回数との比較では、SR 木が最大 $B = 4$ の 59898 点投入後の範囲検索で 19766 回増えているのに対し、EGF では 3336 回の増加となっている。これからも高い動的特性が伺える。

最近傍検索についての考察を行う。MBR 情報を持たない EGF は、ほぼすべての B 値において SR 木と比べてアクセスコストは高い。しかし (MBR を有する) EGF では SR 木に対してほぼ同等かそれ以上であり、実験 1 では最大で $B = 64$ の 100000 点で 1.5 倍、実験 2 では $B = 32$ の 19898 点投入で 2.0 倍のアクセスコストの比率となった。これより、データ内在判定による空バケットの回避もアクセスコスト低減に加え、MBR 情報によるグリッド空間でのアクセスバケットの絞込みが、アクセスコストを低減させる決め手となっていることがわかる。

データが一様分布する場合、データ内在判定や MBR 情報の効果は期待できないが、非一様分布よりもグリッド空間の分割数が小さくなり、空バケットが少なくなることから、完全一致型検索、範囲検索ではほぼ同等のアクセスコスト、最近傍検索ではさらに低いアクセスコストとなると予想できる。

6. 関 連 研 究

現在の空間データ処理システムは大別すると、木構造、近似ファイル、グリッドファイルの 3 つに分類される[3),10)]。

データ分割手法に基づく木構造は、最小包囲長方形 (MBR) に含まれる空間データを階層的な木構造で組織化するものであり、R 木,R*木,SS 木,SR 木などが挙げられる。R*木[1)] は R 木において、体積を小さくするような領域分割戦略を使用したものである。そのため範囲検索には優れるが、ある次元の辺長が大きくなる可能性もあるため、最近傍検索には向かない。SS 木[13)] はノードの形を球としたものである。径を小さくするような領域分割戦略を使用し最近傍検索には優れるが、長方形領域質問の範囲検索では性能が劣化する。A 木[11)] は近似ファイルと木構造の二つを用いる手法である。ノードに対して、子ノードの MBR を仮想包囲長方形 (VBR) と呼ばれる近似コードで相対的に表現するもので、それによりノードの次数を増やすことで検索コストの低減を図るものである。我々の知る限りでは、A 木と SR 木は最近傍検索において最も効率のよい手法である。しかし総じて、R 木と同じように MBR の重なりを許容するため平衡木にはなる

が、完全一致検索や範囲検索でのトラバースの増加やデータ更新の点などで問題を抱えている。

近似ファイルを用いる手法には VA ファイル,A 木が挙げられる。VA ファイル[12] では、データ空間をセルに分割し、各セルにビット列を割り当てるものである。問題点として検索時にはすべての VA ファイルが候補を見つけるために吟味される。また、非一様分布のデータでは近似誤差が大きくなることが知られている。

グリッドファイル[8] は、ファイルの負荷率を考慮して空間を排他的に分割していく非階層的索引構造である。分割された空間はグリッドによって管理される。完全一致検索や挿入操作は最も高速に行える手法であるが、空バケットの存在を許すため最近傍検索のアクセスコストは高くなる、BANG ファイル[2] ではグリッドの重なりを許すことで空バケットを回避しているが、木構造と同様に重複検索の問題が生ずる。

7. 結　　論

本論文では拡張可能グリッドファイルを定義した。そしてその検索手法のアクセスコストを実験により評価し，その有用性を提示した。今後の課題として、膨大なデータ量を想定しひとつの計算機のみに特化せず、複数の計算機による分散環境下での拡張可能グリッドファイルの構築などが挙げられる。

謝　　辞

本論文の執筆に際し、貴重な情報をいただいた桜井保志氏 (NTT サイバースペース研究所), 植村俊亮教授 (奈良先端科学技術大学院大学) に感謝します。

参 考 文 献

1) Beckmann, N. et al.: The R*-tree : An Efficient and Robust Access Method for Points and Rectangles, *SIGMOD* 1990, p.322-331
2) Freeston, M.: The BANG file – A New Kind of Grid File, proc.*SIGMOD* 1987, p.260-269
3) Gaede, V. and Gunther, O.: Multidimensional Access Methods, *ACM Comp. Surveys* 30-2, pp.170-231, 1998
4) A. Guttman: "R-trees: A dynamic index structure for spatial searching" , proc. Int. Conf. ACM SIGMOD '84, pp.47-57,1984
5) T.R. ハーブロン, 遠山元道 (訳): "ファイルシステム" , 啓学出版,1992
6) 片山紀夫, 佐藤真一: "マルチメディア情報の大規模処理に向けた多次元インデクシング手法の応用" , 電子情報通信学会論文誌 (D-II),vol.J82-D-II,no.10,pp.1606-1616,Oct,1999
7) 三好涼介, 三浦孝夫: "拡張可能グリッドファイルによる空間データの検索" DEWS 2004
8) J. Nievergelt , H, Hinterberger: "The Grid File: An Adaptable, Symmetric Multikey File Structure" , ACM TODS. vol. 9, pp.38-71,Mar.1984
9) 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" , 電子情報通信学会論文誌 (D-I),Vol.J74-D-I,No.8,pp.467-475,1991
10) 坂内正夫, 大沢裕: "画像データベース" , 昭晃堂,1987
11) Sakurai,Y., Yoshikawa, M. et al.: The A-tree : An Index Structure for High-Dimensional Spaces Using Relative Approximation, proc. *VLDB* 2000, p.516-526
12) Weber, R., Schek. H.J. et al: A Quantitative Analysis and Performance Study for Similarity-Search Methods in High Dimensional Spaces, proc. *VLDB*, 1998, p.194-205
13) White, D.A. et al.: Similarity Indexing with the SS-tree, IEEE *ICDE*, 1996, p.516-523