# ユビキタスデータのためのインデキシング技術 UBI-Tree における検索コスト見積もり方式の改良

荒川 豊　　中村 隆幸　　中村 元紀

日本電信電話株式会社 NTT 未来ねっと研究所 〒180-8585 東京都武蔵野市緑町 3-9-11

E-mail: {arakawa.yutaka, nakamura.takayuki, nakamura.motonori}@lab.ntt.co.jp

**あらまし**　多様なセンサやアクチュエータが遍在するユビキタス社会においては，流通するデータも多様となり，各データが含む次元（属性）の数や種類は様々なものになる．そこで我々は，R-tree を拡張し，このような多種多様なデータに適した新しいインデキシング方式 UBI-Tree を提案してきた．これまでの UBI-Tree では，各ノードに対応するバウンディングボックスの各次元上の幅を検索時のコストとみなし，これを最小化するようデータ挿入を行っていた．しかしながら，この方式では正しく検索時コストを見積もっておらず，検索効率の低下を招いていた．そこで本稿では，より正確なコスト見積もり方式を提案し，実験により検索効率が改善することを示す．

**キーワード**　多次元インデックス，UBI-Tree，R-Tree，検索コスト

## 1. はじめに

多数のセンサを搭載したスマートフォンの普及，新たなセンサネットワーク技術[1][2]や参加型センシング技術[3]の進展などにより，実世界にセンサやアクチュエータが遍在し，利用可能となりつつある．来るべきユビキタス社会においては，このようなセンサやアクチュエータを含む，多様かつ大量のデバイス端末が通信し合い，様々なユビキタスサービスが実現されると予想される．

現在，我々はこのようなネットワーク上のセンサ／アクチュエータが送受信するデータの配信・蓄積を行うミドルウェアとして，uTupleSpace の研究を進めている[4]．図 1 に uTupleSpace を含むシステム全体の概要を示す．様々なデバイスが uTupleSpace に接続し，多様なデータを蓄積する．また一方で様々なアプリケーションが uTupleSpace に接続し，これらの蓄積データに対して多様なアクセスを行う．我々が想定するこのような状況においては，データ種別を跨る，横断的な多次元範囲検索が有用であり，そうした検索要求が多くなると考えられる．例えば，x，y 方向の 2 次元加速度センサから得られる 2 次元データと，x，y，z 方向の 3 次元加速度センサから得られる 3 次元データが混在して蓄積されている場合，x，y 方向の加速度データを必要とするアプリケーションはどちらの種類のデータも検索できると良い．またセンサデータは多くの場合連続値であるから，範囲検索できることが望ましい．

データベースの分野においては，多次元データに対する範囲検索や近傍検索等を効率化するインデキシング技術の研究が盛んに行われてきた．しかしながら，上記のような，次元の数や種類が異なるデータ集合から横断的な多次元範囲検索を効率的に行う方式は未だ確立されていない．XML-DB や，スキーマレスを謳った DB[5][6]もいくつか存在するが，全文検索技術を利用した完全一致検索や 1 次元の範囲検索に対応するものがほとんどである．そこで我々は，このような次元の数や種類の異なるデータ（異種異数次元データと呼ぶ）集合に対する横断的多次元範囲検索を可能とするインデキシング方式，UBI-Tree を提案した[7]．

しかしながら，これまでの UBI-Tree では，データ挿入時の挿入先ノード選択やノード分割において検索コストの見積もり精度が低く，これにより検索効率が落ちてしまう場合があった．そこで本稿では，この問題に対し，新しい検索コスト見積もり方式の導入により効率改善を行う新しい UBI-Tree の提案を行う．

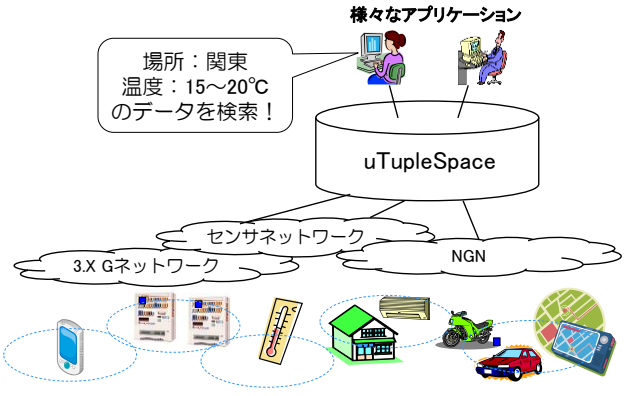


図 1．システム全体概要

## 2. 従来の UBI-Tree とその問題

### 2.1. 従来の UBI-Tree

UBI-Tree は，最もシンプルな多次元インデキシング技術の一つである R-Tree[8] をベースとし，異種異数次元データ集合を扱えるよう拡張を加えたものである．

R-Tree と同様に，UBI-Tree は格納すべき多次元データ集合を最小包囲方形（MBR）と呼ばれる方形領域へ

再帰的に分割する平衡木である．各ノードには，データないし子ノードへのポインタと，それ以下の部分木に格納されている多次元データ集合に対する MBR 情報とを対にして格納する（対となった情報をエントリと呼ぶ）．検索時には，ルートノードから，検索条件範囲と重なる MBR をもつ子ノードのみを辿っていくことで，検索条件に合致するデータを発見することができる．データ挿入時には，ルートノードから，**挿入先子ノード選択アルゴリズム**により選択された子ノードを再帰的に辿り，辿りついたリーフノードへデータを挿入する．これにより，検索実行時のアクセスノード数を削減し，検索効率を高めることができる．ノードが含む子ノードへのポインタ数ないしデータ数には閾値として最小値 $E_{min}$，最大値 $E_{max}$ が設けられており，データ挿入により $E_{max}$ を超えた場合には，ノード分割を行う．この分割は，**ノード分割アルゴリズム**に従って行われる．

R-Tree は，同種同数次元データ集合と同種同数次元の範囲検索を前提としており，上記各アルゴリズムにおいて，各ノードに対応する MBR の超立方体体積を検索コストとし，これを小さくする戦略をとる．しかしながら，異種異数次元データ集合においては，各ノードに対応する MBR の次元数が異なるため，それぞれの超立方体体積を単純に比較することは意味をなさなくなってしまう．また我々が想定する環境においては，各次元が範囲検索条件に用いられる頻度にも大きな差があると考えられる．

そこで UBI-Tree では，ノード間の次元数の違いや各次元が検索条件に用いられる確率を考慮して「各ノードが検索時にアクセスされる確率」を求め，これを小さくすることにより検索効率を向上させる．各アルゴリズムを以下に示す．

**挿入先子ノード選択アルゴリズム：**

挿入しても「検索時にアクセスされる確率」の増加が最も小さい子ノードを選択する．

**ノード分割アルゴリズム：**

分割後の２つのノードの「検索時にアクセスされる確率」の和がなるべく小さくなるよう分割する．

このとき，ノードＮが検索時にアクセスされる確率を，次式で求める．

$$\sum_{S}\left(\left(\prod_{i} N_{dsi}/W_{dsi}\right)\cdot P_c(S)\right) \qquad (1)$$

ただし，S は全次元集合 V の部分集合を表し（S⊆V），次元 $d_{s1}$, $d_{s2}$,…を含む．$N_{dsi}$ はノード N の MBR における次元 $d_{si}$ 方向の幅，$W_{dsi}$ は次元 $d_{si}$ の定義域の幅，また $P_c(S)$は次元集合 S を用いた検索条件で検索される確率を示す．次元 $d_{si}$ を用いた検索条件で検索を行う際，ノード N がアクセスされる確率を $N_{dsi}/W_{dsi}$ で見積もり，V の部分集合を用いた検索条件で検索した場合にノード N がアクセスされる確率の期待値を考えれば，(1)式が得られる．また，$P_c(S)$は次式で求める．

$$P_c(S) = \frac{1}{M}\sum_{T_S}\frac{1}{2^{D_{TS}}-1} \qquad (2)$$

$T_S$ は次元集合 S を含む蓄積データ，$D_{TS}$ はデータ $T_S$ が含む次元数，M は蓄積データ総数を示す．

ある次元をもつデータが 1 万個，別の次元をもつデータが数個蓄積されている場合であれば，前者の次元が検索条件に含まれる確率が高いと予測される． (2)式はそうした予測に基づいた式であり，例えば(x=1), (x=2), (x=3,y=4), (y=5), (x=6, y=7)という 5 つのデータが蓄積されている場合，検索において各検索条件が用いられる確率は(2)式によってそれぞれ次のように見積もられる．

x で検索する確率=(1+1+1/3+1/3)/5=8/15
y で検索する確率=(1/3+1+1/3)/5=5/15
x,y で検索する確率=(1/3+1/3)/5=2/15

## 2.2. 従来の UBI-Tree の問題

従来の UBI-Tree においては，(1)式で得られる値を検索コストとみなし，これがなるべく増加しないようにデータ挿入先の選択やノード分割を行う．あるノードに，それまで当該ノードに存在しなかった新しい次元 dx をもつデータが挿入された場合，通常当該データの値は 1 点であり，$N_{dx}$=0 となる．このとき，(1)式から分かるように，次元の増加は検索コストの増加にはならない．一方，次元 $N_{dx}$ が既存であり，$N_{dx}$ が拡大する場合には検索コスト増加となる．つまり，各ノードには新しい次元を持つデータを挿入しやすく，従来の UBI-Tree では各ノードの次元数は増加しやすいという性質がある．特に，我々が扱うセンサデータは，１つ１つは低次元ながら，全体としては高次元のデータ集合となるため，各ノードが高次元となりやすい．

我々が想定する検索は，多種多様な次元を含むデータ集合の中から，特定の次元と値を含むデータを検索するものである．このような場合，値によるデータの絞込みだけでなく，次元の種類によるデータの絞込みも検索効率の向上に寄与すると考えられる．しかしながら，従来の UBI-Tree では，上記のように各ノードが多種多様な次元を含むようになり，次元種類によるデータの絞込みの効果が働きにくいという問題がある．また次元種類数が膨大になってくると，「次元の呪い」と呼ばれる問題が発生し，木構造構築において最適な

クラスタリングを行うことが本質的に難しくなり，検索効率の低下が発生してしまう．

次元の増加による検索コストの増加を無視する(1)式を採用していることが，このような問題を引き起こす原因と言える．

## 3. 提案方式

本節では，まず一般的なアクセスノード数の期待値の求め方について述べ，次にそれに基づいた UBI-Tree の新しい検索コスト見積もり方式と，その高速化手法について述べる．

### 3.1. アクセスノード数の見積もり

n 次元データ集合に対する検索木において，アクセスノード数の期待値を求める一般的な考え方について述べる[9][10].

「$x=X_{start}\sim X_{end}$」を次元 x の値が $X_{start}$ 以上 $X_{end}$ 以下であるという検索条件を示すこととする．このとき，2 次元平面上に$(X_{start}, X_{end})$をプロットすると，次元 x に対する検索条件は図 2 に示す上三角形内の 1 点に対応付けることができる．上三角形内に絞られるのは，一般に検索条件は $X_{start}\leqq X_{end}$ として良いこと，また通常データの定義域 $X_{min}\sim X_{max}$ に対し，$X_{start}\geqq X_{min}$，$X_{end}\leqq X_{max}$ であることによる．

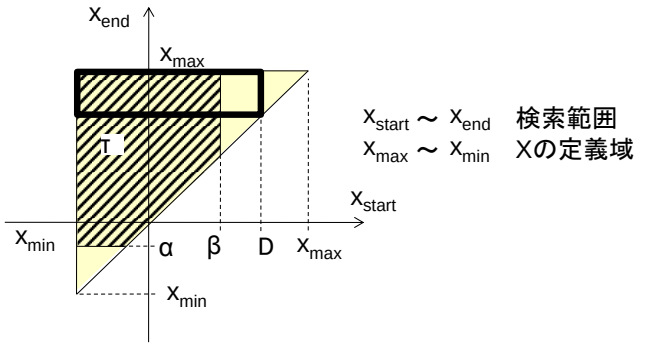


図 2．次元 x に対する範囲検索条件

m 次元の範囲問合せ q は，次のように表現することができる．

$$q: (x_1=X_{1start}\sim X_{1end})\wedge\ldots\wedge(x_m=X_{mstart}\sim X_{mend})$$

上記議論を m 次元に拡張して考えると，問合せ q は，m 個の上三角形の直積空間である 2m 次元空間上の 1 点に対応付けることができる．

木構造を構成するあるノード p の MBR が次元 x について α～β までの領域をもつとすると，図 2 中において斜線で示した領域 τ に含まれる問合せは当該ノードにアクセスする必要がある．つまり，領域 τ の中で問合せが発生する確率が，そのまま当該ノード p にアクセスする確率に影響する．ノード p の MBR が含む x 以外の次元やそれらの組み合わせについても同様の領域を考え，当該領域群の中で問合せが発生する確率を P(p)とすると，検索時にノード p にアクセスする確率は P(p)そのものとなる．木構造を構成するノード群を $p_1\sim p_M$ とすると，アクセスノード数の期待値 C は次式で求められる．

$$C=\sum_{j=1}^{M}P(p_j) \qquad (3)$$

### 3.2. UBI-Tree における新しい検索コスト見積もり方式

2.2 節で述べたように，従来の UBI-Tree においては，あるノードにそれまで存在しなかった新しい次元 dx をもつデータが挿入された場合，通常当該データの値は 1 点であるから，検索コストの増分がないと判断していた．ノードの次元増加は異種異数次元データ特有の現象であり，これにより従来の UBI-Tree においては検索効率の低下が発生していた．

そこで，UBI-Tree において，3.1 節で述べたアクセスノード数見積もりの考え方を導入する．これにより，例えば x 次元上の点 D が新たに挿入された場合，図 2 の太線で囲われた領域分が新たに加わる検索コストとなる．挿入先ノード選択時ないしノード分割時には，「式(3)の増分」が少ないノードあるいは分割方法を選択する．ただし式(3)をそのまま求めるのではなく，挿入や分割によって変化するノード群の $P(p_j)$の増分についてのみ考慮・計算すれば良い．異種異数次元データ集合を扱う UBI-Tree においては各ノードが保持する次元の数や種類が異なるが，ノード分割時に $P(p_j)$ の増分を求める際には，当該ノードに含まれる次元についてのみ計算すれば良い．また同様に，データ挿入時においては，挿入するデータが保持する次元についてのみ計算すれば良い．

次元集合 S を用いた検索条件で検索される確率 $P_c(S)$を用いて，P(p)は次式で与えられる．

$$P(p)=\sum_S P_c(S)\cdot P(p,S) \qquad (4)$$

ここで P(p,S)は，次元集合 S を用いた検索条件で検索した場合に，ノード p がアクセスされる確率である．新しい検索コスト見積もりでは，(1)式に代わり本(4)式を用いる．(4)式について，$P_c(S)$は(2)式で求め，また計算量を抑えるために要素数が 2 以上の S に該当する項を 0 で近似する．実際，要素数が大きい S に対する $P_c(S)$は小さくなりやすい．また要素数が 1 の S に対する P(p,S)は，単純に当該次元における「領域 τ の面積／上三角形の面積」で見積もる．

### 3.3. 新しい検索コスト見積もり方式の高速化

提案する新しい UBI-Tree のデータ挿入／ノード分割アルゴリズムでは，検索コストとして 3.2 節に記述したものを用いる．ただし，ノード分割アルゴリズムにおいて,「検索時にアクセスされる確率」の和がなるべく小さくなる分割方法を探す際，分割方法は組合せの数になるため，総当たり法で探すとノード分割のコストが非常に大きくなってしまう．そのため，最適ではなくとも，なるべく良い分割方法を効率的に見つけ出す近似解法が必要となる．そこで，次に示すアルゴリズムを導入する．

**ノード分割アルゴリズム（NS）:**

NS1. 分割後のノードを 2 つ準備し，それぞれ N1, N2 と呼ぶ．分割対象のノードを N と呼ぶ．

NS2. **初期エントリ選択アルゴリズム**を用いて選択した 2 つのエントリをそれぞれ N1，N2 に挿入する．

NS3.（$E_{min}$－ノード N1 のエントリ数）≧ノード N の未分割エントリ数　であれば，ノード N の未分割エントリを全て N1 に挿入する．

NS4.（$E_{min}$－ノード N2 のエントリ数）≧ノード N の未分割エントリ数　であれば，ノード N の未分割エントリを全て N2 に挿入する．

NS5. ノード N に未分割エントリがあれば，**次回エントリ選択アルゴリズム**を用いて選択したエントリを，同じく選択した挿入先ノードへ挿入し，NS3 へ進む．

**初期エントリ選択アルゴリズム（IE）:**

IE1. ノード N の未分割エントリから,「1 つのノードに挿入した場合に，当該ノードが「検索時にアクセスされる確率」が最も大きくなるエントリペア」を選択する．該当するエントリペアが複数存在する場合は,そのうちの任意の 1 つを選択する．

**次回エントリ選択アルゴリズム（NE）:**

NE1. ノード N の未分割エントリのうち，｜（ノード N1 に挿入した場合の「検索時にアクセスされる確率」の増加量）－（ノード N2 に挿入した場合の「検索時にアクセスされる確率」の増加量）｜が最も大きいエントリを選択する．該当するエントリが複数存在する場合は，そのうちの任意の 1 つを選択する．またこのとき，挿入先ノードとして「検索時にアクセスされる確率」の増加量が小さいノードを選択する．

## 4. 評価

提案方式を導入した新しい UBI-Tree について, 一次評価を行った．

### 4.1. 評価内容

評価対象を(1) R-Tree，(2)従来の UBI-Tree，(3) 新しい UBI-Tree とし, それぞれで疑似データをインデキシングした場合の，Tree 内のレベル（木の高さ）毎のノードの平均次元数と，想定クエリで検索した場合のアクセスノード数を計測した．$E_{max}$, $E_{min}$ はそれぞれ 10, 5 とした．

R-Tree は同種同数次元のデータ集合に対してのみインデキシング可能であるため，R-Tree でインデキシングを行う際には，各データが全種類の次元を保持するよう次元を補完し，インデキシングを行った．補完した次元の値は全て 0 とした．

疑似データは 1 万個，1 データあたり 7 つの次元をもつものとし，次の 2 種類のデータを用いた．データ 1 では各次元がランダムに出現するのに対し, データ 2 は実際のセンサデータを鑑み，センサ種別によって異なる種類のデータが出力されることを模擬したものである．

**データ 1:**

次元種類は全部で 700（a1～a700）とし，それらが各データにランダムに出現する．ただし，1 データ内に同じ次元が複数回出現しない．各次元の値は 0～1000 からランダムに選出した．データ 1 の例を図 3 に示す．

**データ 2:**

次元種類は全部で 700 とし，データ種類数は 100（a1～a100）とする．各データはいずれかのデータ種類に分類され，同じデータ種類のデータは同じ 7 つの次元をもつ（aXX_0～aXX_6）．異なるデータ種類のデータ間では同じ次元をもたない．各次元の値は 0～1000 からランダムに選出した．データ 2 の例を図 4 に示す．

[a698=488　a552=213　a417=978　a179=740　a80=108　a110=298　a671=327]

[a575=698　a525=317　a104=593　a191=443　a302=894　a446=19　a637=789]

[a56=512　a646=100　a617=581　a37=28　a152=130　a641=983　a461=963]

図 3. データ 1 の例

[a37_0=655 a37_1=378 a37_2=640 a37_3=450 a37_4=553 a37_5=83 a37_6=953]

[a21_0=79 a21_1=200 a21_2=353 a21_3=217 a21_4=62 a21_5=372 a21_6=346]

[a37_0=451 a37_1=12 a37_2=796 a37_3=744 a37_4=201 a37_5=864 a37_6=790]

図 4. データ 2 の例

また検索条件は 1000 個，各条件は 3 次元の範囲条件（3 項の AND 条件）とし，次の 3 種類を用いた．データ 1 を検索する場合には検索条件 1 を，データ 2 を検索する場合には検索条件 2－1 または 2－2 をそれぞれ用いた．またアクセスノード数としては，1000 回の検索の総計を計測した．

**検索条件 1：**

700 の次元から，ランダムに 3 つを選択し，条件とする次元とした．値の範囲は，0～1000 からランダムに 2 点選び，小さい方を範囲条件の始点，大きい方を終点とした．検索条件 1 の例を図 5 に示す．

[a257=587~768 a328=373~661 a437=271~990]

[a263=16~154 a611=419~746 a127=238~795]

[a640=358~757 a546=122~190 a697=429~570]

図 5. 検索条件 1 の例

**検索条件 2－1：**

データ 2 の 100 のデータ種類の中からランダムに 1 つ選択し，そのデータ種類に含まれる 7 つの次元からランダムに 3 つを選び，条件とする次元とした．値の範囲は検索条件 1 と同様．検索条件 2－1 の例を図 6 に示す．

[a21_4=19~746 a21_1=641~682 a21_5=278~880]

[a76_1=394~405 a76_3=504~792 a76_5=42~708]

[a61_5=50~948 a61_3=129~864 a61_6=363~413]

図 6. 検索条件 2－1 の例

**検索条件 2－2：**

条件とする次元の選び方は検索条件 2－1 と同様．ただし，値の範囲は一律に 0～1000 とした．検索条件 2－2 の例を図 7 に示す．

[a21_4=0~1000 a21_1=0~1000 a21_5=0~1000]

[a76_1=0~1000 a76_3=0~1000 a76_5=0~1000]

[a61_5=0~1000 a61_3=0~1000 a61_6=0~1000]

図 7. 検索条件 2－2 の例

## 4.2. 評価結果と考察

まず，データ 1，2 について，各 Tree によってインデキシングした場合の，Tree 内のレベル（lv）毎のノードの平均次元数をそれぞれ図 8，9 に示す．lv0 はリーフノードを意味し，lv4 はルートノードを意味する．

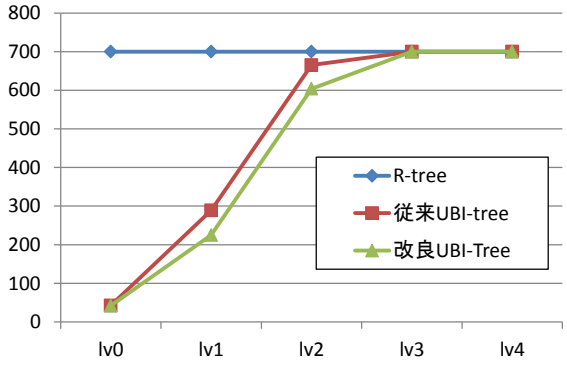


図 8. データ 1 インデキシング時の平均次元数

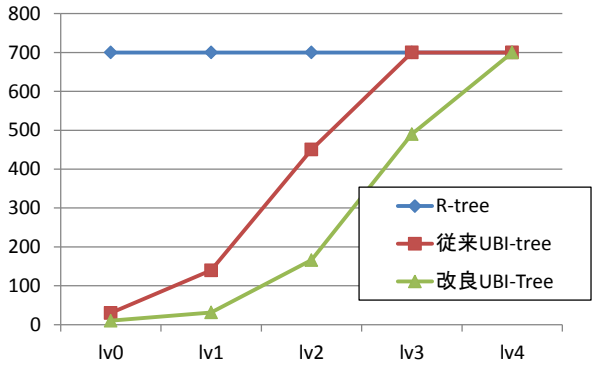


図 9. データ 2 インデキシング時の平均次元数

図 8，9 から分かるように，データ 1，2 のいずれにおいても，R-Tree では，各データが全種類の次元を保持するよう次元を補完してインデキシングを行ったため，リーフノードでも次元数が大きい．これに対し UBI-Tree では，リーフノードに近づくにつれて次元数が減少している．従来 UBI-Tree でも次元減少が見られるが，これは各ノードが含むエントリが 5～10 であるという制約に起因するものである．例えばリーフノードの場合，5～10 個のデータを含み，各データの含む次元が全て異なったとしても，高々70 次元である．

図 8 を見ると，データ 1 では，従来 UBI-Tree に比べ改良 UBI-Tree では多少の次元数減少が見られるものの，大きな差はないことが分かる．これは，各次元がランダムに出現する場合，各ノードの次元数が減るようデータを分類することが本質的に難しいためであると考えられる．

一方，図 9 を見ると，データ 2 では，従来 UBI-Tree

に比べ，改良 UBI-Tree において，大幅に次元が減少していることが分かる．実際のセンサデータの場合には，センサ種別ごとにデータがうまく分類され，このような結果となると考えられる．

データ 1 に対し，検索条件 1 で検索を行った場合の総アクセスノード数を図 10 に示す．また，データ 2 に対し，検索条件 2－1，検索条件 2－2 で検索を行った場合の総アクセスノード数をそれぞれ図 11, 12 に示す．なお，図 12 にはスケールの関係で併記していないが，R-Tree のアクセスノード数は 1997000 であった．

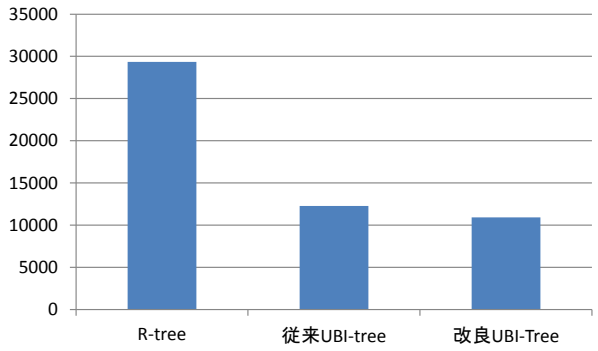


図 10. データ 1 に対する検索条件 1 での検索におけるアクセスノード数

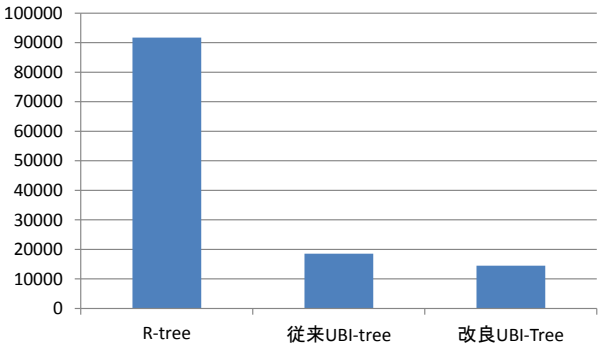


図 11. データ 2 に対する検索条件 2－1 での検索におけるアクセスノード数

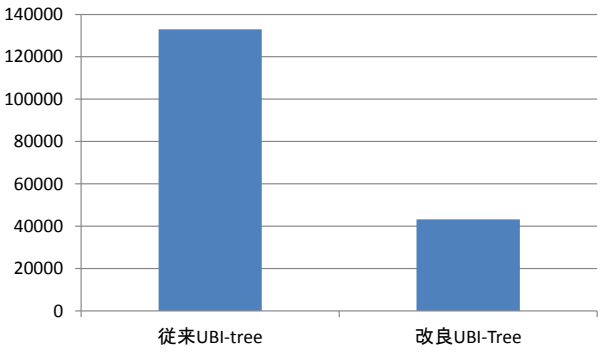


図 12. データ 2 に対する検索条件 2－2 での検索におけるアクセスノード数

図 10, 11 から分かるように，R-Tree に比べ，UBI-Tree を使った検索においてはアクセスノード数が少ないことが分かる．アクセスノード数は検索時の処理量に比例するものであるから，R-Tree に比べ UBI-Tree は優れた検索性能をもつと言える．

図 10 を見ると，データ 1 の場合，従来 UBI-Tree に比べ，改良 UBI-Tree の方が僅かに検索性能が向上していることが分かる．図 8 で確認できるように，データ 1 ではどちらの UBI-Tree においても次元数の減少度合いがあまり変わらないため，検索性能にも大差がないと考えられる．

これに比べて図 11 を見ると，従来 UBI-Tree と比べた場合に改良 UBI-Tree の検索性能向上の度合いがより大きくなることが分かる．図 9 で確認できるように，各 UBI-Tree における次元数減少度合いの違いが，この差を生みだしたものと考えられる．

さらに，図 12 を見ると，検索条件 2－2 で検索した場合，従来 UBI-Tree に比べ，改良 UBI-Tree の検索性能が圧倒的に良いことが分かる．従来 UBI-Tree は，ルートノードからリーフノードまでの次元数の減り方が遅いため，検索条件 2－2 のように範囲条件の幅が大きい場合には，うまくノードを絞り込めないことを示していると考えられる．逆に，改良 UBI-Tree においては，次元数の減り方が速いため，検索条件 2－2 の場合でも，次元種類による絞込み効果により，検索対象とするノードを絞り込めたものと思われる．範囲条件の幅が大きい検索，あるいは「温度情報を含むデータ」（「温度＝－∞～∞」という検索条件とみなすことができる）といった属性の存在のみを指定する検索に対して，提案する UBI-Tree が特に有効であると言える．

## 5. 更なる次元減少手法

本節では，本稿での提案手法と，我々がこれまでに提案してきた改良手法を組み合わせる手法について述べる．

2.2 節で述べたように，従来の UBI-Tree には各ノードの次元数が増加しやすいという問題がある．我々はこの問題に対し，「次元の増加量」を直接的に検索コストとみなし，データ挿入時のノード選択やノード分割を行う手法を提案してきた（「直接法」と呼ぶ）[11]．本稿で提案する検索コスト見積もり方式（「間接法」と呼ぶ）を直接法と組み合わせることで，更に各ノードの次元数を低減することが可能である．

間接法は，各ノードへアクセスする確率を検索コストとし，次元の種類や数だけでなく各次元の値を考慮し，その時点での最適なノード選択やノード分割を行おうとするものである．しかしながら，R-Tree においても同様に問題とされるように，その時点では最適で

も，長期的には不利となる場合がある．特に，本稿のように多種多様なユビキタスデータを扱う場合には，不用意な次元の増加は異種データを同じノードに混在させることに繋がり，検索効率低下の原因となり得る．つまり，ある時点で最適だからといって，ノードの次元数を増加させる選択を行うべきではない場合がある．このような検索効率低下が多発するような場合には，前述のように「直接法」と「間接法」を組み合わせ，次元数の増加を極力抑える手法が有効である．組み合わせた場合の，データ挿入時のノード選択アルゴリズムやノード分割アルゴリズムとしては，例えば以下が考えられる．

**挿入先子ノード選択アルゴリズム：**

1. 「直接法」による検索コスト見積もりにより，挿入先子ノード選択を行う．
2. 1 では挿入先子ノードを一意に絞れなかった場合，1 で候補となったノードについて，さらに「間接法」による検索コスト見積もりを行い，挿入先子ノード選択を行う．

**ノード分割アルゴリズム（概要）：**

1. 「直接法」による検索コスト見積もりにより，最適なノード分割方法を選択する．
2. 1 では最適なノード分割方法を一意に絞れなかった場合，1 で候補となったノード分割方法について，さらに「間接法」による検索コスト見積もりを行い，最適なノード分割方法を選択する．
3. 選択したノード分割方法に従い，ノード分割を実施する．

ノード分割アルゴリズムの詳細としては，初期エントリ選択アルゴリズムや次回エントリ選択アルゴリズムにおいて，それぞれ「直接法」「間接法」をこの順で適用するものが考えられる．

4 節で用いた疑似データ 1，2 を用いて，「直接法」と「間接法」を組み合わせた手法の次元減少効果を確認した結果を図 13，14 に示す．図より，組み合わせた手法では，本稿で提案する「間接法」のみの場合に比べ，各ノードの平均次元数が小さくなることが分かる．

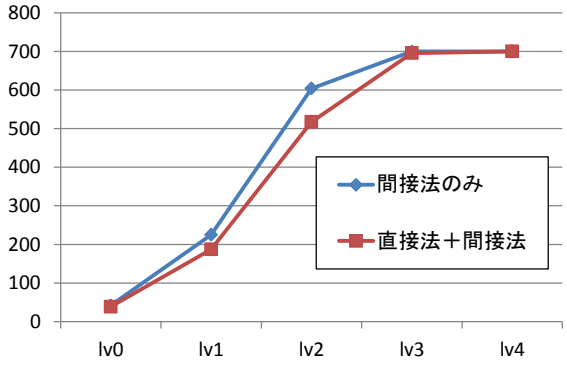


図 13．データ 1 インデキシング時の平均次元数

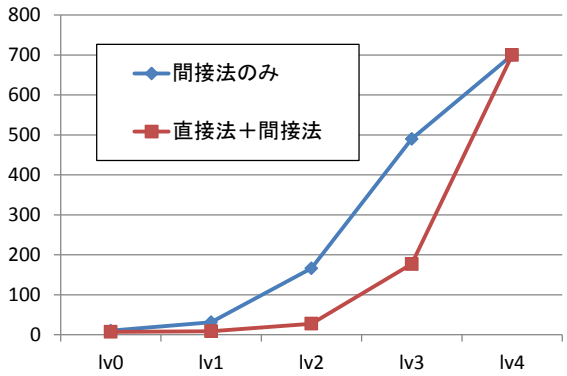


図 14．データ 2 インデキシング時の平均次元数

## 6. 関連研究

現在最も使用されているインデキシング技術は B-Tree（あるいはその派生技術）であろう．B-Tree はよく知られているように 1 次元インデックスであるが，多次元範囲検索に用いられることも多い．その方法の 1 つは，検索条件に含まれる 1 つの次元について B-Tree でインデキシングし，多次元範囲検索時には B-Tree を用いて検索結果の候補を当該次元で絞り込んだ後，さらに他の次元の検索条件を満たすかどうかフィルタリングを行い，最終的な検索結果を得る，というものである．また，複数の次元それぞれについて B-Tree でインデキシングし，多次元範囲検索時には各次元で検索した結果をマージすることで最終的な検索結果を得る，という手法もある．しかしながらこれらの手法は，データが大量になった場合には，フィルタリングやマージの処理コストが大きくなってしまうという問題を有する．

これに対し，UBI-Tree のベースとなっている R-Tree は，B-Tree を多次元化した技術であり，多次元インデキシング技術の中で最もシンプルかつよく知られたものである．多次元範囲検索を効率的に処理できるものの，前述のように同種同数次元のデータ集合に対する

インデキシングが前提となっており，本稿で扱うような異種異数次元のユビキタスデータを効率的に扱うことはできない．

R-Tree には R+-Tree[12]や R*-Tree[13]などの派生技術も多く存在する．これらは MBR の周囲長や重なりなどを加味したコスト見積もり，データの再挿入やエントリ数の下限数制約撤廃などにより，検索効率を向上させるものである．MBR 間の重なりを削減するこれらの改良の多くは，UBI-Tree にも適用できると考えられる．

多次元データの検索においては grid-file[14]など空間分割を行う手法もあるが，ユビキタスデータは全体としては高次元でも，各データが含む次元は少数であるため無駄なグリッドが多くなり，また次元の増加に伴う空間拡張の処理コストも高いと考えられる．

線形探索を行う VA-file[15]は，高次元データ検索において，情報圧縮によるデータ入力量の削減により検索効率を向上させるが，ユビキタスデータのように 1 つ 1 つのデータサイズが小さく，データ数が大量になるような場合には，その効果が小さくなると考えられる．

LDR と呼ばれる手法[16]は，高次元データ集合を局所的に関係のあるデータごとにクラスタ化し，各クラスタに対して次元縮小を行う．特に 5 節で述べた直接法と間接法を組み合わせた UBI-Tree は，まず次元が同じもの同士を集め，その後次元毎の値で分類した木構造となるため，LDR の手法に近い．UBI-Tree では，ユビキタスデータの特性を活かし，より簡単かつシームレスに，次元縮小のクラスタ化とインデキシングを行っていると言える．

## 7. おわりに

本稿では，より精度の高い検索コスト見積もり方式を導入した，新しい UBI-Tree の提案を行った．またその評価の結果，従来の UBI-Tree に比べ，特に範囲条件の幅が大きい検索，あるいは属性の存在のみを指定する検索に対し，良い検索性能を達成することを示した．

今後は，更なる検索性能向上を目指し，従来から R-Tree に対して検討されてきた改善方式を適用するなど，アルゴリズムの改良を行う予定である．また実アプリケーションによる実証実験を通じ，提案手法の有効性について検証を行っていく．

## 参 考 文 献


[1] H. Saito, O. Kagami, M. Umehira and Y. Kado, “Wide area ubiquitous network: the network operator’s view of a sensor network,” IEEE Commun. Mag., vol. 46, no. 12, pp. 112-120, Dec. 2008.

[2] Live E!, “Live E! ～活きた地球の環境情報～,” http://www.live-e.org/

[3] 武山政直, 植木淳朗, “参加型都市センシングによる価値共創モデルの可能性,” 情報処理, Vol. 51, No. 9, pp. 1164-1170, Sep. 2010.

[4] T. Nakamura, M. Nakamura, A. Yamamoto, K. Kashiwagi, Y. Arakawa, M. Matsuo and H. Minami, “uTupleSpace: A Bi-Directional Shared Data Space for Wide-Area Sensor Network,” Proc. of 2nd Intl. Workshop on Sensor Networks and Ambient Intelligence (SeNAmI 2009), pp. 396-401, Dec. 2009.

[5] The Apache Software Foundation, “Apache CouchDB: The Apache CouchDB Project,” http://couchdb.apache.org/

[6] 10gen, Inc., “MongoDB,” http://www.mongodb.org/

[7] 荒川豊，中村隆幸，中村元紀，松尾真人，“UBI-Tree: ユビキタスデータのためのインデキシング技術,” 2010 情処全大, no. 5C-1, pp. 615-616, Mar. 2010.

[8] A. Guttman, “R-Trees: A Dynamic Index Structure for Spatial Searching,” Proc. of ACM SIGMOD, pp.47-57, 1984.

[9] S. Fushimi, M. Kitsuregawa, M. Nakayama, H. Tanaka and T. Motooka, “Algorithm and Performance Evaluation of Adaptive Multidimensional Clustering Technique,” Proc. of ACM SIGMOD, pp. 308-318, 1985.

[10] 大森匡, 佐藤龍生, 星守, “問合せ分布を考慮した R 木における領域分割方式,” 信学論(D-I), vol. J86-D-I, no. 10, pp. 746-761, Oct. 2003.

[11] 荒川豊, 中村隆幸, 中村元紀, 松尾真人, “ユビキタスデータのためのインデキシング技術 UBI-tree の改良,” 信学技報 DE2010-22, 110(162): 47-52, Aug. 2010.

[12] T. Sellis, N. Roussopoulos and C. Faloutsos, “The R+-Tree: A Dynamic Index for Multi-Dimensional Objects,” Proc. of 13th VLDB, pp. 507-518, 1987.

[13] N. Beckmann and H.P. Kriegel. “The R*-tree: An Efficient and Robust Access Method for Points and Rectangles,” Proc. of ACM SIGMOD, pp. 322-331, 1990.

[14] P. A. Burrough and R. A. McDonnell, “Principles of Geographical Information Systems,” Oxford, 1998.

[15] R. Wever, H. -J. Schek and S. Blott, “A Quantitative Analysis and Performance Study for Similarity-Search Methods in High-Dimensional Spaces,” Proc. of 24th VLDB, pp. 194-205, 1998.

[16] K. Chakrabarti and S. Mehrotra, “Local Dimensionality Reduction: A New Approach to Indexing High Dimensional Spaces,” Proc. of 26th VLDB, pp. 89-100, 2000.