Vol.2014-DBS-159 No.6
Vol.2014-IFAT-115 No.6
2014/8/1

複数拠点統合型センサネットワークにおける センシング情報を考慮した時空間インデックス構築手法

川住 涼[1,a)]　義久 智樹[1]　原 隆浩[1]　西尾 章治郎[1]

概要：これまでに複数拠点統合型センサネットワークにおいて，センシング位置を考慮して検索用インデックスを構築することで所望のモバイルセンサデータを持つ拠点を高速に検索する手法が提案されている．しかし，センシング時刻も考慮することで，時刻を指定するクエリにおいても同様に検索を高速に行える．そこで本稿では，複数拠点統合型センサネットワークにおけるセンシング情報を考慮した時空間インデックス構築手法を提案する．提案手法では，各拠点が有するモバイルセンサデータの位置や時刻に関する情報の範囲を木構造を用いてインデックスサーバで管理する．評価の結果，所望のデータを有する拠点を高精度に取得しつつ，従来のインデックス構築手法より検索時間を短縮できることが分かった．

キーワード：センシング, 時空間データベース, インデクシング

1. はじめに

近年，位置や温度といったセンサデータを取得できるスマートフォンなどの小型の移動型端末が急速に普及している．移動型端末から得られるセンサデータはモバイルセンサデータと呼ばれ，多くの人が持つスマートフォン等から周期的にセンサデータを収集することで，広範囲にわたる多数のモバイルセンサデータを収集できる ([1]) ．より多くのモバイルセンサデータを収集するため，複数の組織が運営するセンサデータ収集拠点を連携させて統合的に利用する複数拠点統合型センサネットワークが利用されることがある ([2], [3]) ．例えば，大阪市内の人流把握を行うために，センサネットワーク拠点 A とセンサネットワーク拠点 B から大阪市内のモバイルセンサデータを検索して地図上に表示する際に複数拠点統合型センサネットワークが用いられる．複数拠点統合型センサネットワークの構成を図 1 に示す．各センサネットワーク拠点には管理サーバがあり，モバイルセンサデータの収集や検索などを管理している．各センサネットワーク拠点で収集されたモバイルセンサデータは，各センサネットワーク拠点にあるセンサデータベースに格納される．複数拠点統合型センサネットワークでは，管理サーバが相互にネットワークで接続されている．

[1] 大阪大学大学院情報科学研究科
Graduate School of Information Science and Technology, Osaka University
[a)] kawasumi.ryo@ist.osaka-u.ac.jp

筆者らの研究グループでは，複数拠点統合型センサネットワークにおいて，クエリで指定された地理的範囲内のモバイルセンサデータを持つ拠点を高速に検索するために，センシング位置を考慮して検索用インデックスを構築する手法を提案してきた ([4])．センシング時刻に関する情報も考慮することで，時刻範囲を指定するクエリにおいても検索ができると考えられるが，これまでの手法のほとんどは，位置や時刻が同じデータをまとめて管理することでインデックスの数を削減して検索時間を削減していた．複数拠点統合型センサネットワークでは，センシング時刻が異なることが一般的であり，これまでの手法では効果的にインデックスの数を削減することができない．

そこで本稿では，複数拠点統合型センサネットワークにおけるセンシング情報を考慮した時空間インデックス構築手法を提案する．提案手法では，筆者らが以前に提案した手法を拡張し，複数拠点統合型センサネットワークにおいて，各センサネットワーク拠点がセンサデータベースに格納しているモバイルセンサデータの位置や時刻の範囲を

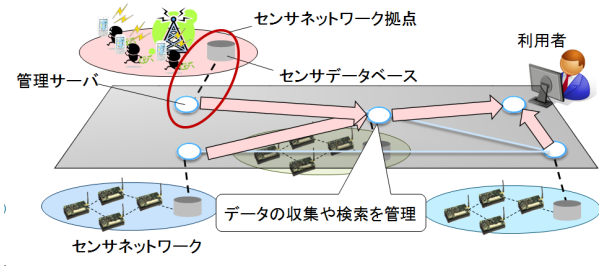

図 1　拠点統合型センサネットワーク

木構造を用いてインデックスサーバで管理する．各センサネットワーク拠点においても，モバイルセンサデータの位置や時刻の範囲を木構造を用いて管理する．クエリで指定される位置や時刻の範囲の大きさを考慮しつつ，時空間の距離的に近い複数の情報をまとめて管理することで，インデックスの数を削減する．まとめすぎるとあるインデックスに多数のセンサデータが含まれて検索に時間がかかるため，Bucket を用いて効率よくまとめる点に新規性がある．さらに，提案手法を実装し，実行速度や精度に関する評価を行った．評価の結果，所望のデータを有する拠点を高精度に取得しつつ，従来の R*-tree[5] を用いたインデックスや，文献 [4] の手法を用いたインデックスによる検索より検索時間を短縮できる場合があることが分かった．

本稿の構成を以下に示す．2 章では，本研究に関連する既存研究を紹介し，本研究との関連について述べる．3 章では，本研究が想定する環境について述べる．4 章では，提案手法について述べる．5 章では，提案手法の性能評価について述べる．最後に 6 章で，まとめと今後の課題について述べる．

2. 関連研究

複数拠点統合型センサネットワークにおける時空間インデックス構築手法の関連研究として，2.1 節で複数拠点統合型センサネットワークに関する研究，2.2 節で空間インデックス構築手法に関する研究をそれぞれ紹介する．2.3 節では，不確実な情報を持つデータを対象とした範囲検索に関する研究について紹介する．

2.1 複数拠点統合型センサネットワーク

GSN (Global Sensor Network, [2]) は，各センサネットワーク拠点の管理サーバで P2P オーバレイを構築している．P2P オーバレイでは，ピア同士がサーバを介さずに直接通信を行うため，サーバへの負荷集中が発生しない．文献 [6] において，GSN を用いてスマートフォンからモバイルセンサデータを収集する研究が行われているが，所望のモバイルセンサデータを有するセンサネットワーク拠点検索のための空間インデックスが構築されておらず，検索に時間がかかる問題がある．

広域に分布する複数センサネットワークの統合利用システムとして X-Sensor2.0 が開発されている ([3])．X-Sensor2.0 も GSN と同様に各センサネットワーク拠点をノードとして P2P オーバレイを構築している．X-Sensor2.0 では，ピア間を移動するプログラムであるモバイルエージェントを用いることで，データを処理しながら検索を行うことが可能である．また，収集の対象となるセンサデータの時間的，空間的な条件およびデータへの処理を記述できる STQL (Spatio Temporal Query Language) を提案している．しかし，X-Sensor2.0 は固定センサネットワーク拠点の統合を想定して設計されており，所望のモバイルセンサデータを有する拠点の検索に時間がかかるという問題がある．

2.2 空間インデックス構築手法

位置情報を持つ空間データに対するインデックス構築手法として，R*-tree[5] がある．R*-tree では，最小方形領域 (Minimum Bounding Box, MBB) を利用したデータ構造であり，空間データの位置情報を地理的範囲で参照する木構造の空間インデックスを構築する．地理的範囲が重複しないようにインデックスを構築しているが，空間データが挿入されるたびに地理的範囲を再計算して木構造を再構築する必要があり，空間データの挿入に時間がかかる問題がある．また，R*-tree では，最小方形領域を 3 次元に拡張することで時空間インデックスを構築することができる．本稿で提案する手法では，葉ノードとして，空間範囲内でのデータの存在を示す識別子である Bucket を用いている点で異なる．

2.3 不確実な情報を持つデータを対象とした範囲検索

文献 [7] では，不確実な位置情報を持ったデータを対象とした範囲検索のためのインデックスの構築手法が提案されている．不確実な位置情報の範囲，クエリとの重なりやすさや範囲内の確率密度に基づいて構築したコストモデルを用いて，四分木内へのデータの効果的な配置とデータの検索を実現している．文献 [4] の手法では，距離的に近い要素を併合する時に，併合後の要素内には一様な確率で実際にデータが存在すると仮定して，要素を併合するか否かの判断している．しかし，文献 [4] の手法では，要素を併合する時にクエリとの重なり方を考慮しておらず，これを考慮することでより効果的に要素の併合を行える．

3. 想定環境

3.1 概要

本研究では，図 2 に示す構成の複数拠点統合型センサネットワークを想定する．1 章で述べたとおり，複数拠点統合型センサネットワークには，異なる組織で運営される複数のセンサネットワーク拠点があり，各センサネットワー

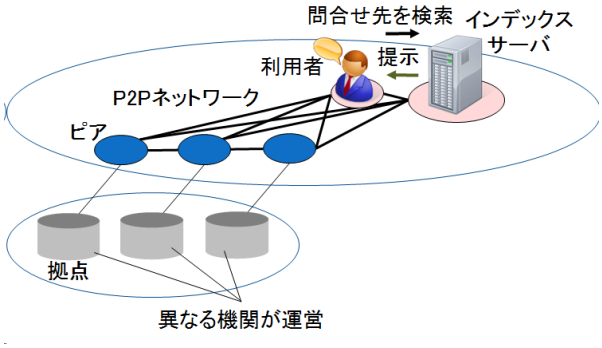

図 2 本研究の想定環境

ク拠点の管理サーバがネットワークで接続されている．異なる組織で運営されており管理責任の都合等から単一の大きなセンサネットワーク拠点として運営できない．また，複数拠点統合型センサネットワークには，モバイルセンサデータの検索時に利用できるインデックスサーバがある．管理サーバやインデックスサーバは，従来の研究 ([2], [3]) においてサーバへの負荷集中が発生しないと確認されている P2P オーバーレイを構築しており，サーバを介さずに直接通信できる．モバイルセンサデータは時空間データであり，緯度と経度の地理情報およびセンシング時刻が付加されている．例えば，北緯 45 度，東経 135 度，6 月 1 日の午前 11 時の温度は 20 度といった温度データや，北緯 46 度，東経 136 度，7 月 20 日の午後 1 時の気圧は 1000hPa といった気圧データが挙げられる．

4. 提案手法

本章では，本研究が提案する複数拠点統合型センサネットワークにおけるモバイルセンサデータの空間インデックスの構築手法の詳細について述べる．

4.1 概要

提案手法では，インデックスサーバ内に時空間インデックスを構築し，各センサネットワーク拠点の管理サーバが Bucket を管理する．時空間インデックスをインデックスサーバがもつことで，すべての管理サーバに問い合わせることなく所望のモバイルセンサデータを取得できる場合が多くなり，検索にかかる時間を短縮できる．また，Bucket に対する操作を，その Bucket に属するモバイルセンサデータを有する管理サーバのみで行うことで，インデックスサーバの更新にかかる送信負荷を抑えられる．

以降，Bucket については 4.2 節で，各センサネットワーク拠点における空間データの管理方法については 4.3 節で，インデックスサーバにおける空間インデックスの構築方法については 4.4 節で説明する．最後に，4.5 節でモバイルセンサデータの検索方法を説明する．

4.2 Bucket

Bucket とは，ある時間的，地理的範囲内における時空間データの存在を示す識別子である．例えば，北緯 45 度，東経 135 度，6 月 1 日午後 1 時の温度データをセンサネットワーク拠点 A が持っており，北緯 46 度，東経 136 度，6 月 1 日午後 2 時の温度データをセンサネットワーク拠点 A と B が持っている場合，6 月 1 日午後 1 時から午後 2 時の範囲内かつ北緯 45 度，東経 135 度から北緯 46 度，東経 136 度の範囲内には，センサネットワーク拠点 A と B の識別子 (IP アドレス等) を含む Bucket が存在する．モバイルセンサデータを検索する際，インデックスサーバから所望の時間的，地理的範囲内に存在する Bucket を発見し，その Bucket に含まれる管理サーバに対してモバイルセンサデータの問合せを行う．

4.3 各センサネットワーク拠点での空間データ管理方法

各センサネットワーク拠点では，モバイルセンサデータを挿入する際に，そのデータの位置が既存の Bucket に含まれている場合にはインデックスの更新を行わず，既存の Bucket に含まれていない場合にのみ，新たな Bucket を作成して，インデックスの更新を行う．初期の Bucket は点領域となる．データの挿入により Bucket の数が多くなりすぎると R*-tree の深さが深くなり，検索や挿入に時間がかかるため，Bucket の重複範囲が大きい場合には Bucket の併合を行う．

4.3.1 モバイルセンサデータの挿入方法

モバイルセンサデータを挿入する際は，文献 [4] で提案した手法と同じく，まず初めに，挿入先の決定を行う．挿入先を決定するアルゴリズムは従来の R*-tree のアルゴリズムと同じものを用いる．次に，挿入したデータが挿入先の兄弟ノードとなる Bucket に含まれるかどうかを調べ，含まれる場合には挿入を行わずに処理を打ち切る．含まれていない場合は，兄弟ノードとの Bucket の併合を試みる (次項で説明) ．併合を行った場合には，併合後の Bucket と残っている兄弟ノードとの併合を試み，これを繰り返す．併合を行った後に，インデックスサーバへのデータの送信を行う．最後に，木構造を維持するために，中間ノードの分割処理や，包含も併合もされなかった Bucket の追加処理を行う．

4.3.2 Bucket の併合

文献 [4] の併合アルゴリズムをセンシング時刻を考慮するように拡張する．このように拡張した場合においても，新しく挿入された Bucket と既存の Bucket を併合する際，併合前の領域と併合後の領域の差が大きいほど，実際のデータ分布との誤差が大きくなり，検索の精度に影響を及ぼす．そこで，領域の差をエラー差分 ΔE (図 3) と定義し，併合する際の基準として利用する．$overlap(V_1, V_2)$ は領域 V_1 と V_2 が重なっている領域の体積である．単純に領域 V_1 と V_2 のエラー差分を定義した場合，エラー差分 ΔE は $\Delta E = V_1 + V_2 - overlap(V_1, V_2)$ と表される．しかし，同一端末から発生するモバイルセンサデータ間には時間的な重なりが生じないため，ΔE の値が大きくなり，検索精度を維持する場合に Bucket の併合が生じにくくなる．そこで，クエリで指定される位置や時刻の範囲の大きさを考慮して V_1，V_2 を拡張することを考える．併合した後に，実際のデータ分布と異なる領域を図 4 に示す．このとき，クエリで指定された範囲内に，実際のデータ分布と異なる領域のみを含む場合に，提案方式は誤った検索結果を返す．従って，データ分布と異なる領域とデータ分布と一致する領域の両方を含む場合は検索結果に影

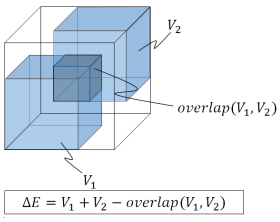

図 3　エラー差分 ΔE

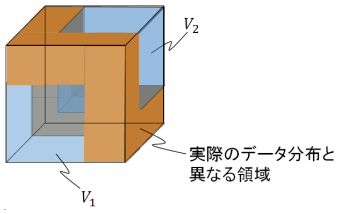

図 4　実際のデータ分布と異なる領域

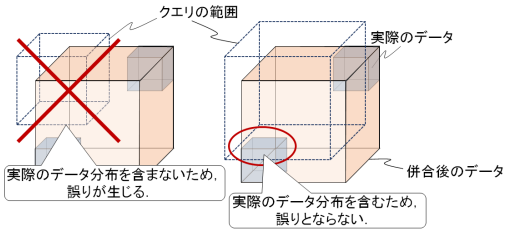

図 5　検索結果に誤りが生じる場合と生じない場合

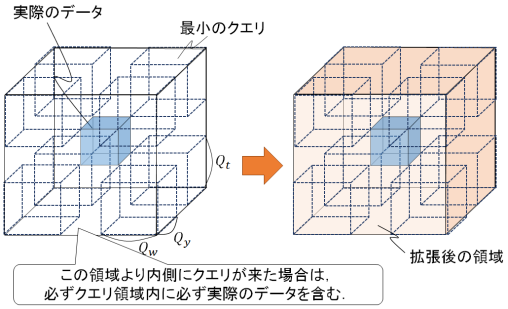

図 6　クエリの指定する範囲の大きさを考慮した Bucket の拡張

響を与えない (図 5)．クエリで指定されうる範囲のうち最小の範囲を持つクエリを Q の，各次元の幅をそれぞれ Q_w, Q_h, Q_t とした場合，図 6 のように Bucket を拡張することで，検索精度の低下を抑えつつ Bucket を拡張できる．併合後の Bucket の大きさに影響を与えないようにするために，これらの拡張を併合後の Bucket の内側に対してのみ行う．V_1 に対する Bucket の拡張処理を expand(V_1) と表すと，エラー差分 ΔE は $\Delta E = expand(\mathrm{V}_1) + expand(\mathrm{V}_2) - overlap(expand(\mathrm{V}_1), expand(\mathrm{V}_2))$ と表すことができる．併合後の Bucket の体積を V_{merged} としたときに，しきい値 E_j に対して，$\frac{\Delta E}{V_{merged}} \leq E_j$ を満たす場合に Bucket の併合を行う．併合先の候補が複数存在する場合は，併合後の体積が最大となる Bucket と併合する．併合を行う場合には，既存の Bucket の領域を拡張する．

Bucket 併合のアルゴリズムを Algorithm 1 に示す．入力は新しく挿入する Bucket$B$ と挿入先のノード C である．C の i 番目の子要素は C.child.get(i) で取得できる．クエリの範囲の大きさを考慮して拡張した Bucket と併合後の Bucket は，getBucketToMerge 関数と getMergedBucket 関数によってそれぞれ取得できる．また，Bucket の体積と 2 つの Bucket が重なっている領域の体積は，getArea 関数と getOverlap 関数によってそれぞれ取得できる．まず初めに，それぞれの兄弟ノードに対してエラー差分を計算する．このとき，getBucketToMerge 関数によってそれぞれの Bucket を拡張する (13 行目〜14 行目)．その後，getMergedBucket 関数によって併合後の Bucket を取得し，取得した Bucket と getOverlap 関数を用いてエラー差分を計算する (15 行目〜16 行目)．エラー差分と併合後の体積に基づいて併合先を更新する (17 行目〜20 行目)．最後に，併合先が存在する場合には併合処理を行う．getMergedBucket 関数で併合後の Bucket を取得した後に一方の Bucket を更新し，インデックスサーバに送信したときの領域の体積を更新する (25 行目〜26 行目)．併合が終わると，古い Bucket の削除を行う (27 行目〜29 行目)．

Algorithm 1 Bucket の併合

```
 1: merge(B, C):
 2: /* 初期化 */
 3: tarea ⇐ 0
 4: tobj ⇐ null
 5: exist ⇐ −1
 6: i ⇐ 0
 7: /* 併合先の Bucket を探索 */
 8: while i < c do
 9:    if C.child.get(i).id == B.id then
10:       exist ⇐ i
11:       continue
12:    end if
13:    v1 ⇐ getBucketToMerge(B,C.child.get(i))
14:    v2 ⇐ getBucketToMerge(C.child.get(i),B)
15:    mbr ⇐ getMergedBucket(v1,v2)
16:    err ⇐ getOverlap(v1,v2) / getArea(mbr)
17:    if tarea < getArea(mbr) && err ≤ E_j then
18:       tarea ⇐ getArea(mbr)
19:       tobj ⇐ C.child.get(i)
20:    end if
21:    i ⇐ i + 1
22: end while
23: /* 併合処理 */
24: if tobj != null then
25:    tobj ⇐ getMergedBucket(B, tobj)
26:    tobj.s ⇐ B.s + tobj.s
27:    if exist > 0 then
28:       C.child.remove(exist)
29:    end if
30:    return tobj
31: end if
32: return null
33: end
```

4.4　インデックスサーバにおける空間インデックスの構築

各センサネットワーク拠点は，インデックスサーバに挿入や更新を行った Bucket の情報を通知する．インデックスサーバは，更新された Bucket の情報を用いて R*-tree を構築し，モバイルセンサデータへのインデックス (モバイルセンサデータが含まれる Bucket を担当するセンサネットワーク拠点管理サーバの識別子) を作成する．文献 [4] で提案した手法と同じく，Bucket に含まれるインデックスに変更がなくても木が再構築される頻度を少なくするため，

Bucket の領域が大きくなってセンサネットワーク拠点の担当範囲が広くなったときのみ再構築を行う．

4.5 インデックスサーバにおける問い合わせ処理

問い合わせ処理では，問い合わせに含まれる地理的範囲と重複する部分がある MBR を順次探索する．文献 [4] で提案した手法と同じく，MBR 内の Bucket と問い合わせに含まれる地理的範囲に重なりがある場合にデータが存在すると判定し，Bucket に含まれるセンサネットワーク拠点 ID を問い合わせ先に加える．ただし，問合せ先として同じ拠点 ID を既に保持している場合は，問合せ先への追加は行わない．

5. 性能評価

提案手法の有用性を確かめるためにデータの検索に関する性能評価を行った．本章ではその方法と結果について述べる．

5.1 評価手法

性能評価を行うために，Java を用いて提案する時空間インデックス構築手法を実装した．性能評価には，CPU が Intel(R) Core(TM) i7-3770 CPU @ 3.40GHz，メモリが 8GB の CentOS 6.3 で動作する計算機を用いた．

また，本評価では，モバイルセンサデータとして，実際に得られたジオタグ付きのツイートデータと，一様乱数によって決定した位置情報および時刻情報を付加したデータの 2 種類を用いた．ジオタグ付きのツイートデータは，2011 年 10 月 24 日から 2012 年 5 月 31 日までに得られた 25853434 個のツイートである．なお，一様乱数で与える時刻情報の範囲はジオタグ付きツイートデータと同様の範囲になるように設定した．一様乱数で得られるデータとは異なり，ジオタグ付きツイートなどの実データでは都市部などにデータが集中するため，併合処理が活発に行われると考えられる．しきい値 E_j は 0.1，各センサネットワーク拠点で発生するモバイルセンサデータの数は 10000 個とし，1000 個のデータ発生毎にインデックスサーバを用いて検索を行い，その性能を評価した．

5.2 評価結果

5.2.1 Bucket の併合アルゴリズムの評価結果

Bucket の併合アルゴリズムに関する性能評価として，検索にかかる時間と検索精度を計測した．問合せは，ユーザが指定した時間的，地理的範囲内にモバイルセンサデータを有するセンサネットワーク拠点を検索することを想定した．変化の詳細については各評価の項で述べる．地理的範囲の指定は正方形により行い，正方形の位置や大きさを変えながら測定した．また，時間的範囲の開始地点や幅も変えながら測定した．検索精度に関しては，問合せの範囲内に含まれるモバイルセンサデータを有するセンサネットワーク拠点を正解とし，正解の結果に対する適合率（問合せ結果の中に含まれる正解の割合）と再現率（正解集合のうち，問合せ結果が占めている割合）を評価指標として用いた．比較手法には，従来の R*-tree，Bucket の併合は行うが，クエリの指定する範囲の大きさを考慮しない方式 (w/o considering query size) を用いた．

評価結果を以下の図 7-14 に示す．

図 7-8 はデータセットにジオタグつきツイートデータを用い，クエリが指定する最小の地理的範囲を 2m 四方の正方形，最小の時間的範囲を 15 分に設定した場合の評価結果である．クエリの指定する範囲の大きさを考慮することによって，検索精度を維持しながらインデックスの数をより削減できるため，検索精度を維持しつつ，検索にかかる時間を削減できる．

図 9-10 はデータセットにジオタグつきツイートデータを用い，クエリが指定する最小の地理的範囲を 200m 四方の正方形，最小の時間的範囲を 1 時間に設定した場合の評価結果である．図 7 と図 9 を比較すると，検索にかかる時間が増加しているが，これは処理するクエリの指定する範囲が大きいため，問合せあたりの探索範囲が大きくなるためである．しかし，クエリの指定する範囲の最小値が大きくなっているため，より Bucket の併合が活発に行われるようになり，検索にかかる時間の削減量は大きくなる．

図 10-14 はデータセットに一様乱数データを用いた場合の評価結果である．データが地理的，時間的に分散しているため，削減量は小さくなるが，インデックスの数を最も多く削減している提案方式が最も良い性能を示す．

5.2.2 センシング時刻の考慮による影響の評価結果

センシング時刻の考慮による影響に関する性能評価として，検索にかかる時間と検索精度を計測した．問合せ，正解の定義に関しては，5.2.1 節と同様の想定で評価を行っ

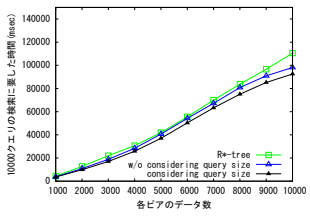

図 7 探索範囲が小さい場合の検索時間 (ツイートデータ)

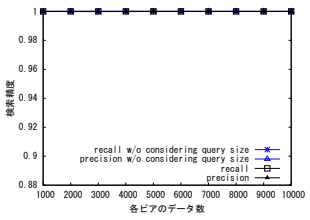

図 8 探索範囲が小さい場合の検索精度 (ツイートデータ)

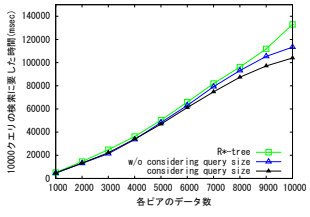

図 9 探索範囲が大きい場合の検索時間 (ツイートデータ)

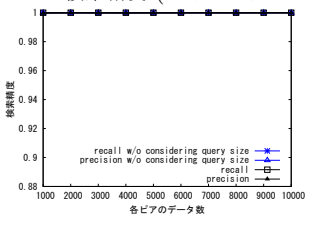

図 10 探索範囲が大きい場合の検索精度 (ツイートデータ)

た．ただし，比較手法には，従来の R*-tree と，提案手法と同様の手法で地理的情報のみを扱う手法を用いた．

評価結果を以下の図 15-18 に示す．proposal method-2d は提案手法と同様の手法で地理的情報のみ扱う手法，proposal method-3d は提案手法を表す．図 15-16 はデータセットにジオタグつきツイートデータを用いた場合の評価結果である．実データでは，Bucket の併合が活発に行われ，かつ時間に関する情報を無視しているため，地理的情報のみを扱う手法のインデックスの数が最も少なくなり，最も検索時間を削減できる．しかし，時間に関する情報を無視しているため，検索精度は提案方式よりも低くなる．

図 17-18 はデータセットに一様乱数データを用いた場合の評価結果である．このデータセットでは各データが地理的にも分散しているため，時間に関する情報を考慮しなくても，Bucket の削減量は少なくなる．このような場合では，検索の際に時間的範囲で参照する要素を絞り込めるため，提案方式が最も良い性能を示している．検索精度に関しても，ツイートデータを用いた場合と同様，地理的情報のみを扱う手法は提案手法に比べて検索精度が低くなる．

6. おわりに

本稿では，複数拠点統合型センサネットワーク上で，モバイルセンサデータを効率的に検索するための時空間インデックスの構築手法を提案した．提案するシステムを Java で用いて実装し，性能評価を行った結果，提案手法に基づいて構築されたインデックスサーバに問合わせることにより，従来のインデックス構築手法に比べて高い精度を維持しつつ高速に問合せ先を取得可能であることを確認した．

今後の課題として，Top-k クエリなど，より具体的なクエリ処理を考慮した拡張を考えている．

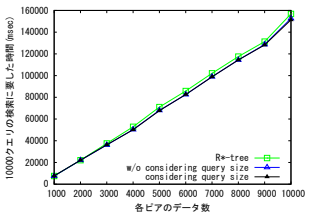

図 11 探索範囲が小さい場合の検索時間 (乱数データ)

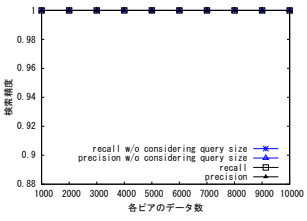

図 12 探索範囲が小さい場合の検索精度 (乱数データ)

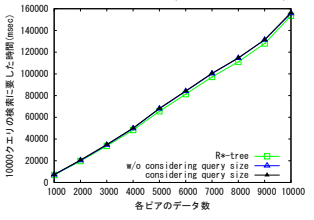

図 13 探索範囲が大きい場合の検索時間 (乱数データ)

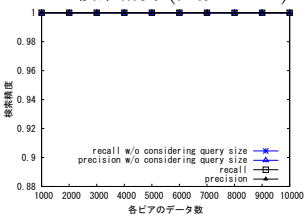

図 14 探索範囲が大きい場合の検索精度 (乱数データ)

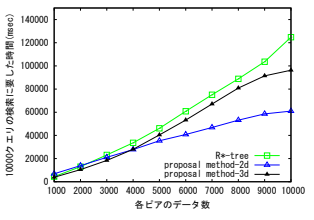

図 15 検索時間 (ツイートデータ)

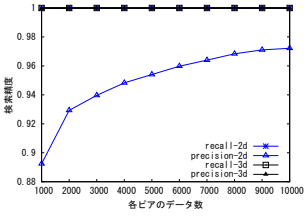

図 16 検索精度 (ツイートデータ)

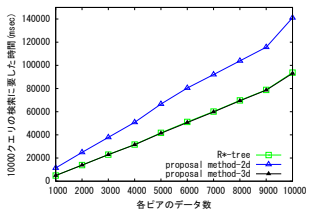

図 17 検索時間 (乱数データ)

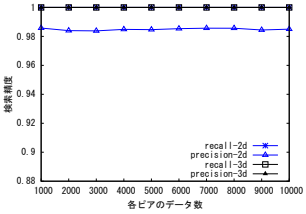

図 18 検索精度 (乱数データ)

謝辞

本研究の一部は，文部科学省国家課題対応型研究開発推進事業－次世代 IT 基盤構築のための研究開発－「社会システム・サービスの最適化のための IT 統合システムの構築」(2012 年度～2016 年度)，科学研究費補助金基盤研究 (A) (課題番号：26240013)，科学研究費補助金若手研究 (A) (課題番号：23680007)，総務省戦略的情報通信研究開発推進事業 (SCOPE) の研究助成による成果である．ここに記して謝意を表す．

参考文献

[1] Wu, F.-J., Lim, H. B., Pereira, F., Zegras, C. and Ben-Akiva, M.: A User-centric Mobility Sensing System for Transportation Activity Surveys, *Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems*, SenSys '13, pp. 74:1–74:2 (2013).

[2] Aberer, K., Hauswirth, M. and Salehi, A.: Global sensor networks, Technical report (2006).

[3] Yoshihisa, T., Hamaguchi, Y., Ishi, Y., Teranishi, Y., Hara, T. and Nishio, S.: A Sensor Data Aggregation System Using Mobile Agents, *Distributed Networks: Intelligence, Security, and Applications*, pp. 39–63 (2013).

[4] 川住　涼，義久智樹，原　隆浩，西尾章治郎：複数拠点統合型センサネットワークにおけるモバイルセンサデータの空間インデックス構築手法，*DEIM Forum 2014 E8-4* (2014).

[5] Beckmann, N., Kriegel, H.-P., Schneider, R. and Seeger, B.: *The R*-tree: an efficient and robust access method for points and rectangles*, Vol. 19, No. 2, ACM (1990).

[6] Perera, C., Zaslavsky, A., Christen, P., Salehi, A. and Georgakopoulos, D.: Capturing sensor data from mobile phones using global sensor network middleware, *Personal Indoor and Mobile Radio Communications (PIMRC), 2012 IEEE 23rd International Symposium on*, pp. 24–29 (2012).

[7] Zhang, Y., Zhang, W., Lin, Q. and Lin, X.: Effectively indexing the multi-dimensional uncertain objects for range searching, *Proceedings of the 15th International Conference on Extending Database Technology*, pp. 504–515 (2012).