DEIM Forum 2010 F3-1

# 距離索引を用いた逆最遠傍問題に対する効率的な検索手法

劉　健全† 陳　漢雄† 古瀬　一隆† 北川　博之†,††

† 筑波大学大学院システム情報工学研究科 〒305–8577 茨城県つくば市天王台 1–1–1
†† 筑波大学計算科学研究センター 〒305–8577 茨城県つくば市天王台 1–1–1
E-mail: †ljq@dblab.is.tsukuba.ac.jp, ††{chx,furuse,kitagawa}@cs.tsukuba.ac.jp

**あらまし**　本論文では，逆最遠傍 (Reverse Furthest Neighbors, RFN) という新たな問合せ問題に対する独自の検索手法を提案する。データセット $O$ 及びクエリ $q$ を与えられたとき，RFN 検索は $O$ の中から $q$ を最遠傍とする全てのオブジェクトを求める。従来手法としては，RFN 問題の凸包特性を利用して，R-tree に基づくアルゴリズムが提案されている。しかし，凸包の計算量は高い。本研究では，計算量を削減するため，ピボットを利用する距離索引を用いた効率的なアルゴリズムを提案する。さらに，合成データと実データを用いて提案手法の効率とスケーラビリティを検証する。

**キーワード**　逆最遠傍 (RFN)，問合せ処理，距離索引，最近傍検索

## 1. はじめに

過去十年間，類似検索は科学研究・開発への応用が急増している。例えば，パターン認識や [1], [2]，画像検索 [3]，時系列マッチング [4] などが挙げられる。これらの応用に伴って，類似検索に関する研究は活発にされている。類似検索に関する問い合わせ処理は，k 近傍 (k-Nearest Neighbors, k-NN) や，範囲問い合わせ (Range Query)，逆最近傍 (Reverse Nearest Neighbors) などの変形である。これらに対するオンライン問い合わせ処理の要求が急増しており，様々な処理技術が提案された。それらは主に，空間分割法 (例：grid-file [5])，データ索引法 (例：R-tree [6])，連続スキャン法 (例：VA-file [7]) との三種類に分けられる。

しかし，近傍の反対，すなわち，遠傍に対する問い合わせ処理はこれまで今後の課題とされ，特に逆最遠傍 (Reverse Furthest Neighbors, RFN) の問い合わせ処理は未着手となっていた。しかし，実世界の応用では，効率的な逆最遠傍の問い合わせ処理は有意義である。Yao たち [8] は，初めて逆最遠傍問題に関する正式な定義を示し，R-tree に基づく問い合わせ処理方法をともに提案した。あるデータセット $O$ と任意のクエリ $q$ を与えられたとき，$O$ の中から $q$ を最遠傍とする全てのオブジェクトの集合を，$q$ の逆最遠傍 (RFN) と言う。また，この問題を解くことを逆最遠傍検索と言う。RFN 問題は，*monochromatic reverse furthest neighbors* (MRFN) 問題 (i.e., RFN) と *bichromatic reverse furthest neighbors* (BRFN) 問題に分けられる。BRFN 問題では，問い合わせ対象には任意のクエリ $q$ ではなく，クエリ集合 $Q$ を与え，ある $q \in Q$ を指定する。BRFN 問い合わせは，クエリ集合 $Q$ の $q$ 以外のオブジェクトとの距離と比べて，$q$ を最遠傍とする全てのオブジェクト $o \in O$ を求めることである。

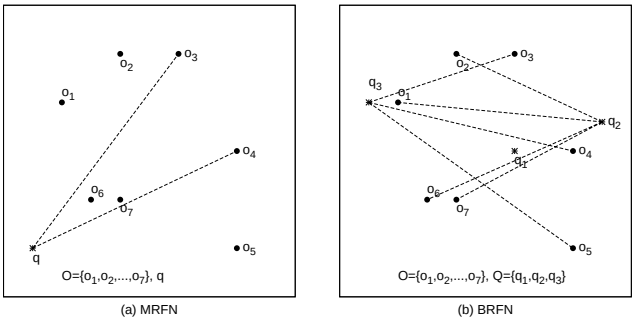


図 1　問い合わせ例 (a) MRFN, and (b) BRFN.

例として，図 1 を用いて説明する。(a) は，MRFN の問い合わせ処理を示している。オブジェクト集合 $O$ 及びクエリ $q$ を与えられたとき，$q$ の逆最遠傍は集合 $\{o_3, o_4\}$ になる。単純な方法でこれを求めるには，各オブジェクト $o_i$ において，$o_i$ と $q$ の距離，及び $o_i$ と他の $o_j (i \neq j)$ の距離を算出し，$o_i$ は $q$ を最遠傍とするかどうかを確認すればよい。$o_3$ と $o_4$ のみが $q$ をそれぞれの最遠傍とするので，$q$ の逆最遠傍の解になることが分かる。

また，図 1 の (b) を用いて BRFN の問い合わせ処理を説明する。データセット $O$ は (a) と同様である。クエリ集合として $Q = \{q_1, q_2, q_3\}$ を与えるとする。そして，各 $q \in Q$ の BRFN を計算してみると，例えば $q_2$ が指定された場合には，$q_2$ 以外の $q \in Q$ と比べ，$q_2$ の BRFN の解は集合 $\{o_1, o_2, o_6, o_7\}$ に

なる。同様に，$q_3$ の BRFN の解は集合 $\{o_3, o_4, o_5\}$ になるが，$q_1$ の BRFN の解は空となる。$O$ の中では，$q_1$ 以外の $q \in Q$ と比べて，$q_1$ を自分の最遠傍とするオブジェクトは存在しないことが分かる。

次に，逆最遠傍問題を求めるのが有意義であることを示すため，　[8] に基づき，以下に具体的な応用例を示す。

**応用例１：**　逆最遠傍問題は観光産業において応用することができる。例えば，東京都にある全ての観光地をデータセット $O$ とする。観光地のあたりにある様々なお店に対して，普段，観光客はその観光地に近い店を利用する。特別な理由がなければ，より遠くの店へは行かない。店の経営者の立場からすれば，遠くてもより多くの顧客に来店してほしいと考える。したがって，経営者にとっては，自分の店を最遠傍 (あるいは観光客が最も行きたくない) とする観光地において，店の宣伝の力を入れるべきである。この場合，宣伝をするべきの観光地はこの店 ($q$) の逆最遠傍となる。

**応用例２：**　都市計画の分野でも応用例が挙げられる。例えば，自治体の都市計画部門では，建築申請を審査する際，市民への影響や自治体の計画などを含めていろいろ検討する。例えば，ある化学工場を建築する提案がなされたときに，自治体が一番考慮するのは，その化学工場の建築候補地は住民にどのぐらい影響するかということである。もちろん，影響しないあるいいは影響の最も少ない候補地のほうが建築する場所として好ましい。このような候補地の選択は，逆最遠傍問題によって行うことができる。この例の場合，建築候補地と住宅との距離が遠ければ遠いほど，住民への影響が少ないと言える。化学工場の建築候補地集合を $Q$ として，住宅の所在地集合を $O$ とすると，候補地 $q \in Q$ の逆最遠傍になる $o \in O$ の数が一番多い $q$ は最適な候補地となる。

**応用例３：**　さらに，逆最遠傍問題は市場経済における戦略分析の面でも応用できる。例えば，PC メーカーにとっては，会社の利益を上げるためによいマーケッティング戦略を実施するのは当然である。自社の製品と顧客との売買関係を分析しながら戦略対策を制定することが必要となる。ここで，顧客の PC メーカーの製品に対する評判を定量的に距離計算ができれば，評判が悪ければ悪いほど，距離が長くなると仮定できる。この際，顧客群を $O$，PC メーカーの集合を $Q$ として BRFN 問題を解くと，自社に対する評価が低い顧客を求めることができる。このような顧客に向け特別な対策を実施することによって売上の向上を目指すといった戦略が採用できる。

本論文では，逆最遠傍問題に対して, [8] で提案された方法より効率的な検索手法を提案する。本論文の構成は以下の通りである。第 2 章で関連研究を紹介する。第 3 章では，本研究の提案手法の基本定理と補題を詳しく説明する。続いて，第 4 章で本研究で用いた距離索引構造及びアルゴリズムを提案し，第 5 章で評価実験によって本提案手法が効率向上であることとスケーラビリティを示す。最後に，本論文での内容を簡単にまとめ，今後の課題を述べる。

## 2. 関 連 研 究

類似検索に関する技術はこれまでにもよく研究されていた。本研究と最も関連のある問い合わせタイプは主に 3 種類に分かられる。すなわち，(k-) 最近傍検索 (NN, あるいは k-NN)，逆 (k-) 最近傍検索 (RNN, あるいは RkNN)，逆最遠傍 (RFN) である。

**最近傍検索** については，早期に R-tree [6] を用いる深さ優先探索法 [9], [10] と最良優先探索法 [11] が提案された。近年，Athitsos たちは距離に基づくハッシュ手法を用いた近似最近傍検索を提案した [12]。また，Tao たちは LSB-tree による高次元データにおける最近傍検索を提案した [13]。

**逆最近傍検索** に関しては，Korn と Muthukrishnan が初めて逆最近傍の問い合わせを提案した [14]。それ以降，様々な問い合わせ処理手法が提案されておいる。例えば，RdNN-Tree [15]，TPL [16]，MRkNNCop-Tree [17] などがある。これらには，大量なデータに対する RNN 応用例も示されている。

**逆最遠傍検索** について，本研究と最も関連するものは [8] である。Yao たちは初めて逆最遠傍検索及びそのクエリタイプ，MRFN と BRFN を定義した。これらの問い合わせを処理するため，彼らは Progressive Furthest Cell(PFC) と Convex Hull Furthest Cell(CHFC) とのアルゴリズムを提案した。これらのアルゴリズムでは，R-tree とともに，最遠ボロノイセル (furthest Voronoi cell, fvc) を利用して，あるデータオブジェクト $o \in O$ が $q$ の RFN になるかどうかを判断する。ここで，$fvc(q, O)$ は，データセット $O$ における，クエリ $q$ に関する凸多辺形を表す。$fvc(q, O)$ を計算するには，$q$ と $o(o \in O)$ の全てのペアの垂直二等分線を引いて，データ空間を 2 つ部分空間に分けて，$q$ と離れている部分空間をそれぞれの交差を取ったものがクエリ $q$ に関する凸多辺形領域になる。この領域が $q$ の逆最遠傍になる $o \in O$ を厳密に囲むことが証明されている。この性質を利用するため，PFC アルゴリズムでは，R-tree における走査をしながら，$fvc(q, O)$ を構築する。同時に，$fvc(q, O)$ を用いて，各 MBR あるいはデータオブジェクトが解になるかをチェックする。しかし，このアルゴリズムでは，解にならないものはそのまま捨てられない。解にならないものを用いて，より狭い $fvc(q, O)$ 領域を再構築することは可能であるため，この後処理は非常に非効率的である。PFC を改善するため，彼らはさらに CHFC アルゴリズムを提案した。CHFC では，逆最遠傍問題に対する特有な凸包特徴を示し，この特徴をさらに利用して $fvc(q, O)$ を構築することで，より効率のよい検索を行う。CHFC アルゴリズムを図 2 に示す。CHFC アルゴリズムは PFC より速いが，効率の悪い処理も実際には含

```
Algorithm: CHFC(Query q; R-tree T)
1  Compute C_P with T using either the distance-priority
       or the depth-first algorithm;
2  if q ⊂ C_P then return ∅;
3  else {
4      Compute C_{P*} using C_P ∪ {q};
5      Set fvc(q, P*) equal to fvc(q, C_{P*});
6      Execute a range query using fvc(q, P*) on T;
7  }
```

図 2　CHFC algorithm.

まれている。例えば，1 と 4 行目に凸包を計算することと，6 行目にある R-tree を用いた範囲問い合わせを実行することは計算コストが高い。これらの計算コストの高さは実験結果の図 5 の (a) より見られる。これらの高価な計算コストを削減するため，本研究では新たなアプローチを提案する。効率向上のための目標は，1) 凸包の計算を避けたること，2)R-tree を使った範囲問い合わせの計算を避けること，3) 検索するときに高価な距離計算をなるべく削減すること，である。

## 3. フィルタリングの定理・補題

本章では，第 2 章に述べたチャレンジに向け，本研究のアプローチを提案するため，基になる定理と補題を詳しく説明する。まず，逆最遠傍検索に関する特有な性質は [8] に幾つかの補題で示されたが，本稿では定理 1 で簡単にまとめる。定理 1 に関する完全な証明について, [8] で確認ができる。

［定理 1］ データセット $O$ 及びその凸包 $C_O$ と，任意のクエリ $q$ を与えられたとき，$q$ の RFN の解が空ではないための必要十分条件は，$q$ が凸包 $C_O$ の外側あるいは境界線にあることである。

定理 1 に沿って，単純に効率的なフィルタリングだけで，R-tree を用いた範囲問い合わせを避けることが可能になる。同時に，フィルタリングの効果で解にならないものは既に排除されたので，大量な距離計算を省くことも可能になる。フィルタリングを行うため，本章では，定理 1 を基に次の補題 1 と補題 2 を定義する。

定義を分かりやすくするため，まず，定理・補題をフィルタリングで利用する動機を説明する。データセット $O$ から凸包 $C_O$ の全ての頂点を選んでピボットセット $S_{piv}$ とすると仮定する。定理 1 により，ピボット $p \in S_{piv}$ に対する逆最遠傍の解は存在する可能性がある。この時に，解となるのは $p$ を自分の最遠傍とする全てのオブジェクト ($o \in O$) である。逆に考えると，あるオブジェクト $o \in O$ に対して，$S_{piv}$ が $O$ の凸包であるので，$o$ の最遠傍になるものは，$S_{piv}$ にある一つのピボット $p_i$ である。この場合，オブジェクト $o$ は必ず $p_i$ の逆最遠傍になる。したがって，あるクエリが与えられた時に，もし，あるオブジェクト $o \in O$ とクエリ $q$ の距離 $dist(o,q)$ が，このオブジェクト $o$ の最遠傍のピボット $p_i \in S_{piv}$ との距離 $max_{p_i \in S_{piv}}\{dist(o,p_i)\}$ より小さければ，このオブジェクト $o$ は $q$ の逆最遠傍になることができない。なぜなら，$o$ は $q$ が自分の最遠傍とはならないからである。一方，距離 $dist(o,q)$ が距離 $max_{p_i \in S_{piv}}\{dist(o,p_i)\}$ より大きければ，このオブジェクト $o$ は必ず $q$ の逆最遠傍になる。理由は，データセット $O$ 中の $o$ の最遠傍は (凸包の頂点である)$S_{piv}$ に存在する可能性があるが，$o$ との最遠距離は $o$ と $q$ の距離 $dist(o,q)$ より小さいので，$S_{piv}$ のピボットは $o$ の最遠傍とはならず，$q$ が $o$ の最遠傍になる。したがって，$o$ は $q$ の逆最遠傍の一つである。このようにフィルタリングをすれば，データセット $O$ にある大量な解にならないものを効率的に排除することができ，解になるものもすぐ決められる。

ここまで，R-tree を用いた範囲問い合わせを行わず，効率的なフィルタリングをするのは可能であることが分かった。しかし，フィルタリングを行うために距離の比較をするには，大量な距離計算 ($dist(o,q)$ と $dist(o,p_i)$) が必要である。これは効率低下な処理であるため，このような大量な距離計算を減らさなければならない。本稿ではこの距離計算を大量に削減するための 2 つの手法を提案する。オブジェクト $o$ と $p_i$ との距離を計算する代わりの手法として，第 4 章において距離索引を提案する。各オブジェクト $o$ に対する距離 $max_{p_i \in S_{piv}}\{dist(o,p_i)\}$ をあらかじめ索引として作っておく。さらに，$o$ と $p_i$ の距離索引もあらかじめ作っておく。一方，オブジェクト $o$ と $q$ の距離計算を削減するには，既存の距離索引を利用して，さらに上下界距離に収束できるような三角不等式に基づいて，距離の比較を行う。これに関する具体な補題を述べる前に，簡単な図 3 を用いて説明する。(a) において，$\triangle qop_1$ について考えると，三角不等式により，オブジェクト $o$ とクエリ $q$ との距離 $dist(o,q)$ の上界は $dist(p_1,q) + dist(o,p_1)$ となる。この上界はオブジェクト $o$ と最も遠いピボット $p_3$ との距離 $dist(o,p_3)$ より小さいので，$o$ は $q$ の逆最遠傍の解にならないから，対象外ととして安全に除外することができる。同様に，(b) では，三角不等式により，オブジェクト $o$ とクエリ $q$ との距離 $dist(o,q)$ の下界は $|dist(p_1,q) - dist(o,p_1)|$ となる。この下界は $dist(o,p_3)$ より大きいので，$o$ の最遠傍は $p_3$ ではなく，代わりに $q$ を最遠傍とする。言い換えれば，この場合，$o$ は必ず $q$ の逆最遠傍の一つになる。

次に，前述の動機に基づく補題を証明とともに示す。

［補題 1］ データセット $O = \{o_1, \cdots, o_n\}$ が与えられたとき，その凸包を $C_O$ とする。$C_O$ の全ての頂点からなる集合をピボットセット $S_{piv} = \{p_1, \cdots, p_m\}$ とする。あるクエリ $q$ が与えられたとき，任意のオブジェクト $o \in O$ に対して，下記の不等式 1 を満たせば，$o$ は $q$ の逆最遠傍にはなり得ない。

$$min_{p_i \in S_{piv}}\{dist(o,p_i) + dist(p_i,q)\} < max_{p_i \in S_{piv}}\{dist(o,p_i)\} \quad (1)$$

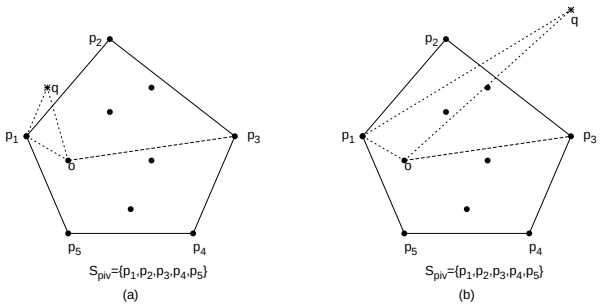


図 3　補題を示す例：(a) 補題 1，(b) 補題 2.

［証明 1］ 図 3 で示すように，三角不等式により，下記の不等式が成り立つ。

$$dist(o,q) \leqq dist(o,p_i) + dist(p_i,q),$$

$o$ と $q$ との距離が未知である場合でも，効率化のため，この距離の計算は避けたい。代わりに，$o$ と $p_i$ との距離が索引に存在しており，$p_i$ と $q$ との距離はすぐ求められるので，全ての距離の和の組み合わせ $dist(o,p_i)+dist(p_i,q)$ に対して，下記の式から $o$ と $q$ の距離の上界を求めることができる。

$$U_{bound}(dist(o,q)) = min_{p_i \in S_{piv}}\{dist(o,p_i) + dist(p_i,q)\}.$$

一方，不等式 1 の右辺は $o$ と最も遠いピボット $p \in S_{piv}$ との距離を表す。言い換えれば，$o$ は $p$ の逆最遠傍の一つである。よって，もし不等式 1 は成り立てば，下記の不等式が成り立つ。

$$U_{bound}(dist(o,q)) < max_{p_i \in S_{piv}}\{dist(o,p_i)\}$$

ここで，$o$ と $q$ との距離の上界は $o$ と最遠のピボット $p$ との距離より小さいので，$o$ は $q$ の逆最遠傍になり得ない。証明終わり。

［補題 2］ データセット $O = \{o_1, \cdots, o_n\}$ が与えられたとき，その凸包を $C_O$ とする。$C_O$ の全ての頂点からなる集合をピボットセット $S_{piv} = \{p_1, \cdots, p_m\}$ とする。あるクエリ $q$ が与えられたとき，任意のオブジェクト $o \in O$ に対して，下記の不等式 2 を満たせば，$o$ は必ず $q$ の逆最遠傍の一つになる。

$$max_{p_i \in S_{piv}}\{|dist(o,p_i) - dist(p_i,q)|\} \quad (2)$$
$$> max_{p_i \in S_{piv}}\{dist(o,p_i)\}$$

［証明 2］ 補題 1 と同様に，三角不等式により，下記の不等式が成り立つ。

$$dist(o,q) \geqq |dist(o,p_i) - dist(p_i,q)|$$

ここで，全ての距離の差の組み合わせ $|dist(o,p_i)-dist(p_i,q)|$ に対して，下記の式から $o$ と $q$ の距離の下界を求められる。

$$L_{bound}(dist(o,q)) = max_{p_i \in S_{piv}}\{|dist(o,p_i) - dist(p_i,q)|\}.$$

同様に，もし不等式 2 が成り立てば，下記の不等式が成り立つ。

$$L_{bound}(dist(o,q)) > max_{p_i \in S_{piv}}\{dist(o,p_i)\},$$

ここで，$o$ と $q$ との距離の下界は $o$ と最遠のピボット $p$ との距離より大きいので，$q$ は $o$ の最遠傍になり，逆に $o$ は必ず $q$ の逆最遠傍の一つになる。証明終わり。

## 4. 距離索引とアルゴリズム

前述の定理と補題に基づいて，検索中に高速なフィルタリング処理を行うことができる。ただし，補題 1 と補題 2 を用いるには，データセット $O$ の凸包の全ての頂点をピボットセット $S_{piv}$ として選択し，これらのピボットをさらに利用して，逆最遠傍の解にならないものを排除し，解になるものをピックアップする。したがって，補題で使った距離をあらかじめ計算し索引に置かなければならない。このため，本稿では次の 2 種類の距離索引を提案する。

### 4.1 距 離 索 引

データセット $O$ にある各オブジェクト $o$ の最遠傍となるピボットとの距離 $max_{p_i \in S_{piv}}\{dist(o,p_i)\}$ をハッシュテーブル **MetricIndex-A** に保存する。各オブジェクトの ID$o_{id}$ をキーとし，最遠ピボットとの距離 $max_{p_i \in S_{piv}}\{dist(o,p_i)\}$ を値とする。ピボットセット $S_{piv}$ とデータセット $O$ のペア毎の距離について，実際に，図**??**のような巨大な行列を保存しなければならないが，本提案では，代わりにハッシュテーブル **MetricIndex-B** を用いて，キー・値のマッピングを保存する。そこで，ID ペア $(p_{id}, o_{id})$ をキーとして，これらの距離をハッシュテーブルへマップして保存する。ハッシュテーブルを利用することで，オブジェクト間の距離を読み取るのは $O(1)$ のコストで済む。

MetricIndex-A

| $o_{id}$ | $max_{i=1}^{m}\{dist(o,p_i)\}$ |
|---|---|
| 1 | $d_1$ |
| 2 | $d_2$ |
| ⋮ | ⋮ |
| n | $d_n$ |

MetricIndex-B

| | $o_1$ | $o_2$ | $\cdots$ | $o_n$ |
|---|---|---|---|---|
| $p_1$ | $d_{11}$ | $d_{12}$ | $\cdots$ | $d_{1n}$ |
| $p_2$ | $d_{21}$ | $d_{22}$ | $\cdots$ | $d_{2n}$ |
| ⋮ | ⋮ | ⋮ | ⋱ | ⋮ |
| $p_m$ | $d_{m1}$ | $d_{m2}$ | $\cdots$ | $d_{mn}$ |

図 4　距離索引のハッシュテーブル

### 4.2 アルゴリズム

第 3 章で述べた定理と補題，及び前述の距離索引を用いて，逆最遠傍問題に対する効率的なアルゴリズムを提案する。このアルゴリズムの略称を **PIV**) と呼ぶ。PIV アルゴリズムでは，フィルタリング (2-17 行目)，リファインメント (18-25 行目) の 2 つの処理段階に分かれている。フィルタリングでは，全ての処理は本稿で述べた定理と補題に従う。例えば，2 行目は定理 1 によってクエリ $q$ が凸包の中に入るかどうかを

```
Algorithm: PIV( Query q, Pivots S_piv, Dataset O )
1  Initialize candidate set Cand ← ∅, answer set Answ ← ∅;
2  if (is_query_inside_convex_hull(q, S_piv) == TRUE) {
3      return ∅;
4  }
6  Compute each distance pair dist(q, p), p ∈ S_piv;
7  foreach o ∈ O {
8      if (min_{p_i∈S_piv}{dist(o,p_i) + dist(p_i,q)}
           < max_{p_i∈S_piv}{dist(o,p_i)}) {
9          // Discard o by Lemma 1;
10 } else {
11     if (max_{p_i∈S_piv}{|dist(o,p_i) − dist(p_i,q)|}
           > max_{p_i∈S_piv}{dist(o,p_i)}) {
12             Insert o into Answ; //o is answer by Lemma 2;
13         } else {
14             Insert o into Cand; //objects need further check;
15         } //end if
16     } //end if
17 } //end foreach
18 foreach o ∈ Cand {
19     Compute real distance dist(o,q);
20     if (dist(o,q) < max_{p_i∈S_piv}{dist(o,p_i)}) {
21         Discard o;
22     } else {
23         Insert o into Answ;
24     } //end if
25 } //end foreach
26 return Answ;
```

判断する。7 行目から 17 行目までで，データセット $O$ のオブジェクトをスキャンしながら，フィルタリングを行う。補題 1 を満たすオブジェクトは捨てて，満たさなければ，さらに補題 2 に基づく判断をする。満たせば，オブジェクトを解集合 $Answ$ へ追加する。これでも満たさないと，解候補集合 $Cand$ へ入れて，次のリファインメントで改めて処理する。リファインメントでは，$Cand$ に残ってあるオブジェクトを 1 個ずつ見て，クエリ $q$ との実距離を計算して，$q$ の逆最遠傍になるかどうかをチェックする。7 行目，8 行目，11 行目の処理により，PIV アルゴリズムの計算のコストは $O(mN)$ となる。

## 5. 実験と評価

本論文の提案手法の有効性を確認するため，実験を行った。実験環境は以下の通りである。

- CPU: Intel(R) Xeon (R) 2.83 GHz
- 主記憶: 16 GB
- 実装言語: C++

実験データには，合成データと実データを用いた。合成データは一様分布 (UN) とランダムクラスタ分布 (RC) との 2 種類のデータを使った。実データはアメリカの道路ネットワーク図 (Map) を利用した。(注1).

上記の環境で合成データと実データのそれぞれについて，関連研究 [8] の計算方法 CHFC と，力まかせ探索 BFS と，本研究のアルゴリズム PIV との比較実験を行った。計算コスト及び IO コストを比較項目として評価を行った。以下にその結果を示す。

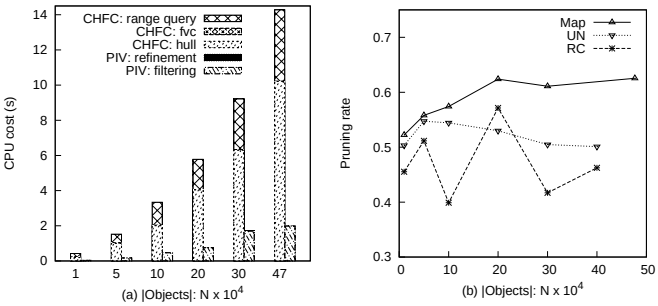


図 5　(a) CHFC vs. PIV. の CPU コスト; (b) PIV の排除率.

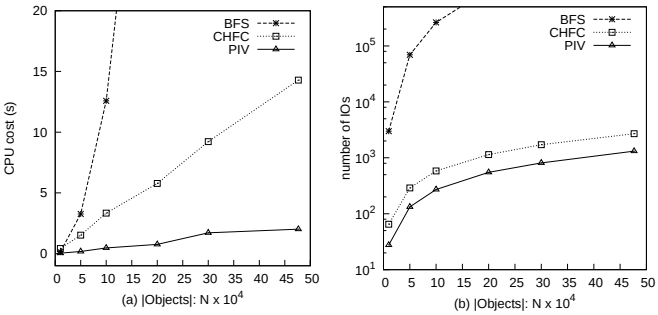


図 6　実データにおける (a)CPU コスト，(b)IO コスト.

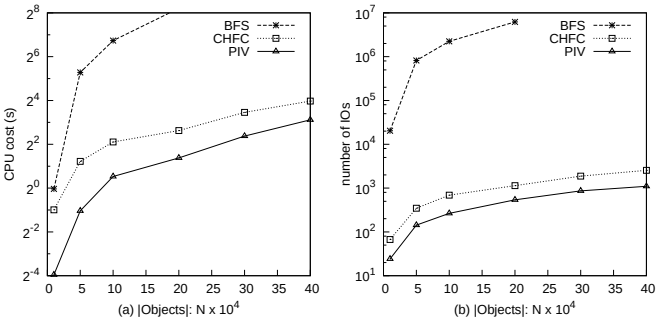


図 7　合成データ (RC) における (a)CPU コスト，(b)IO コスト.

まず，図 5 の (a) は，実データにおいて，CHFC 法の高価な計算部分と，PIV 法の 2 つ計算段階のコストとの比較を示す。ここで，CHFC 法では，凸包の計算と R-tree を使った範囲問い合わせの計算は全体コストの約 90% を占めるている。この結果は，第 2 章に指摘した問題の実証となっている。さらに，結果を見ると，CHFC 法の高価な 2 つ計算部分の

(注1)：http://www.cs.fsu.edu/~lifeifei/SpatialDataset.htm

コストは，いずれも PIV 法の全体コストより高いことが分かる。同時に，図 5 の (b) は，実データと合成データにおいて，PIV 法のフィルタリング段階に解にならないものを排除する性能を示している。実データ (Map) に対して，排除率は 60%を超える。合成データについて，(RC) と (UN) データは排除率がそれぞれ 50%，40%を超える。よって，PIV 法のフィルタリング段階は解にならないものの解除率が高いことが分かった。

また，CHFC 法及び BFS 法との比較をして，図 6 と図 7 より本研究の提案手法の効率とスケーラビリティを評価した。図 6 では，実データを用い，図 7 では，合成データ (RC) を用いた。それぞれのデータにおける計算コストと IO コストを示す。結果によって，逆最遠傍問題に対して，本提案の PIV 法が最も速い問い合わせ処理をすることが実証された。CHFC 法に比べると，処理速度で約 10 倍，IO コストで約 5 倍の向上が見られた。さらに，データセット $O$ のサイズの変化により，CHFC 法と BFS 法より，本提案の PIV 法のほうがより安定的であることが分かった。

## 6. ま と め

本論文では，逆最遠傍 (RFN) という新たな問合せ問題に対する独自な検索手法を提案した。従来手法で提案された RFN 問題の凸包特性を利用して，本稿の基本定理・補題をまとめるともに，ピボットを利用する距離索引を提案し，効率向上な検索アルゴリズムを示した。理論分析により，従来手法に存在する高価な計算コストの問題を解決し，大幅な計算量を削減することを実現した。さらに，合成データと実データを用いて提案手法の効率とスケーラビリティを検証しながら，従来手法より顕著に速くなることを示した。

今後の課題として，本提案手法をさらに汎用的なものとするため，I/O コストの考慮も含めた手法の改良や拡張などを行うとともに，より詳細な評価実験の実施を行うことを検討している。

**謝辞** 本研究の一部は科学研究費補助金特定領域研究 (＃21013004) による。

### 文 献

[1] Serge Belongie, Jitendra Malik, and Jan Puzicha. Shape matching and object recognition using shape contexts. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(4):509–522, 2002.

[2] Kristen Grauman and Trevor Darrell. Fast contour matching using approximate earth mover's distance. In *CVPR (1)*, pages 220–227, 2004.

[3] Ritendra Datta, Dhiraj Joshi, Jia Li, and James Ze Wang. Image retrieval: Ideas, influences, and trends of the new age. *ACM Comput. Surv.*, 40(2), 2008.

[4] Rakesh Agrawal, Christos Faloutsos, and Arun N. Swami. Efficient similarity search in sequence databases. In *FODO*, pages 69–84, 1993.

[5] Jürg Nievergelt, Hans Hinterberger, and Kenneth C. Sevcik. The grid file: An adaptable, symmetric multikey file structure. *ACM Trans. Database Syst.*, 9(1):38–71, 1984.

[6] Antonin Guttman. R-trees: A dynamic index structure for spatial searching. In *SIGMOD Conference*, pages 47–57, 1984.

[7] Roger Weber, Hans-Jörg Schek, and Stephen Blott. A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. In *VLDB*, pages 194–205, 1998.

[8] Bin Yao, Feifei Li, and Piyush Kumar. Reverse furthest neighbors in spatial databases. In *ICDE*, pages 664–675, 2009.

[9] Nick Roussopoulos, Stephen Kelley, and Frédéic Vincent. Nearest neighbor queries. In *SIGMOD Conference*, pages 71–79, 1995.

[10] King Lum Cheung and Ada Wai-Chee Fu. Enhanced nearest neighbour search on the r-tree. *SIGMOD Record*, 27(3):16–21, 1998.

[11] Gísli R. Hjaltason and Hanan Samet. Distance browsing in spatial databases. *ACM Trans. Database Syst.*, 24(2):265–318, 1999.

[12] Vassilis Athitsos, Michalis Potamias, Panagiotis Papapetrou, and George Kollios. Nearest neighbor retrieval using distance-based hashing. In *ICDE*, pages 327–336, 2008.

[13] Yufei Tao, Ke Yi, Cheng Sheng, and Panos Kalnis. Quality and efficiency in high dimensional nearest neighbor search. In *SIGMOD Conference*, pages 563–576, 2009.

[14] Flip Korn and S. Muthukrishnan. Influence sets based on reverse nearest neighbor queries. In *SIGMOD Conference*, pages 201–212, 2000.

[15] Congjun Yang and King-Ip Lin. An index structure for efficient reverse nearest neighbor queries. In *ICDE*, pages 485–492, 2001.

[16] Yufei Tao, Dimitris Papadias, and Xiang Lian. Reverse knn search in arbitrary dimensionality. In *VLDB*, pages 744–755, 2004.

[17] Elke Achtert, Christian Böhm, Peer Kröger, Peter Kunath, Alexey Pryakhin, and Matthias Renz. Efficient reverse k-nearest neighbor search in arbitrary metric spaces. In *SIGMOD Conference*, pages 515–526, 2006.