レコードデータに対するリッジ問合せの性能評価

柿木 健† 北川 博之‡ 天笠 俊之‡

†筑波大学 大学院システム情報工学研究科
‡筑波大学 計算科学研究センター

†‡〒305-8573 茨城県つくば市天王台 1-1-1
E-mail: †t-kakinoki@kde.cs.tsukuba.ac.jp, ‡{kitagawa, amagasa}@cs.tsukuba.ac.jp

あらまし ある集合の中から少なくとも一つの属性で優位になるタプルを探索する処理をスカイライン問合せという．その発表以来，現在に至るまで効率的なアルゴリズムや様々な発展問題が研究されてきた．本研究では，二つの集合を比較して，一方の集合に対して優位なタプルを識別することによって，集合の持つ傾向を把握したいという問題を考える．例えば，事業展開の提携先となる不動産業者を選定する際，物件の価格・最寄駅からの距離等の基準で他の業者と比較して優位な保有物件を調査し,その業者が保有する物件の大まかな特徴(特徴的な物件)を把握したい場合がそれに相当する．ある業者では他の業者と比べて「高価だが最寄駅に近い物件」を保有している傾向があり，別の業者では他の業者と比べて「安価だが最寄駅から遠い物件」を保有している傾向があるかもしれない．本研究では，そのような他の集合と比較して相対的に優位なタプルを探索する問題を「リッジ」問題と定義し，解法のための四つのアルゴリズムを提案する．また，二つの集合の密度，相対的な位置関係，距離等を変化させる実験により，それらのアルゴリズムの効率を検証して特徴を明らかにする．

キーワード リッジ問合せ，スカイライン問合せ，性能評価

Performance Evaluation of Ridge Queries for Record Data

Takeshi KAKINOKI† Hiroyuki KITAGAWA‡ and Toshiyuki AMAGASA‡

†Graduate School of Systems and Information Engineering, University of Tsukuba
‡Center for Computational Sciences, University of Tsukuba

†‡1-1-1 Tennohdai, Tsukuba, Ibaraki, 305-8573 Japan
E-mail: †t-kakinoki@kde.cs.tsukuba.ac.jp, ‡{kitagawa, amagasa}@cs.tsukuba.ac.jp

Abstract Given a set of records, skyline query retrieves a set of records which dominate other records with regard to at least one attribute. Since its proposal to date, a lot of research efforts have been made to improve the performance, and to extend its problem domain. In this paper we attempt to apply it to a class of decision-making problem; given a couple of record sets, we want to retrieve such records in a set that are not dominated by records in the other set. In this way, we are able to observe the tendency of the dataset. For example, when choosing a realtor with which we have to tie up for business development, we would like to investigate dominant properties, owned by the realtor, with some criterions like price and distance from the nearest station, in order to observe the tendency of the properties. One realtor may have cheaper properties but far from the nearest station, while another may have expensive properties but more close to the nearest station. In this paper we propose the “ridge” problem, in which we retrieve relatively dominant records in a dataset compared to the other dataset. We propose four algorithms to compute such problems, and make the characteristics clear by some experiments, in that ensity, relative position, distance are changed.

Keyword Ridge Query, Skyline Query, Performance Evaluation

1. まえがき

ある集合の中から少なくとも一つの属性で優位になるタプルを探索する処理をスカイライン問合せという．その発表以来，現在に至るまでより効率的なアルゴリズムや様々な発展問題が研究されてきた.しかし，意思決定問題はそのような場合だけではなく，二つの集合を比較して複数属性で相対的に優位なタプルを識別し，集合の持つ傾向を把握したい場合がある．例えば,事業展開の提携先となる不動産業者を選定する際，物件の価格・最寄駅からの距離等の基準で他の業者と比較して優位な保有物件を調査し，その業者が保有する物件の大まかな特徴（特徴的な物件）を把握したい

場合がそれに相当する．ある業者では他の業者と比べて「高価だが最寄駅に近い物件」を保有している傾向があり，別の業者では他の業者と比べて「安価だが最寄駅から遠い物件」を保有している傾向があるかもしれない．

そのような他の集合と比較して相対的に優位なタプルを探索する問題を先行研究[3]では「リッジ」問題として定義し，Naïve 及び Skyline という二つのアルゴリズムを提案した．Naïve とはリッジの定義と同様に，リッジを求める対象の集合 P の全点と比較対象の集合 Q の全点を順次比較する手法である．Skyline とは比較対象の集合 Q のスカイラインを先行して求め，その点と集合 P を比較する手法である．

しかし，先行研究[3]で Skyline に組み込んでいるスカイラインを求めるための BNL (Block Nested Loop)という手法は最も基本的なものであり，その他により効率的な手法が存在している．また，[3]の中で行っている実験は集合間の相対的な位置関係を考慮しておらず，集合間の距離の変化も実験しているとは言い難い．さらに，Naïve や Skyline は集合の分布の相違に着目して効率的にリッジを探索するアルゴリズムではなく，リッジ問題固有の効率的手法を提示しているとは言い難い．

そこで本研究では，全スカイラインを求める手法の中で理論上最も効率的な D&C (Divide & Conquer)という手法[1][2]を Skyline の中に組み込んで実験を行う．また，集合間の距離が大きくなっている際に効率的にリッジを探索できる Zoom-in というアルゴリズムを提案し，二つの集合の密度，相対的な位置関係，距離等を変化させる網羅的な実験を行うことにより，Naïve・Skyline-BNL（BNL 組込）・Skyline-DC（D&C 組込）・Zoom-in のアルゴリズムの効率と特徴を検証する．

第 2 節では関連研究を示し，第 3 節ではリッジの定義，及び Zoom-in アルゴリズムの提案を行う．第 4 節では実験対象アルゴリズム，及び実験内容と考察について説明する．

2. 関連研究

スカイライン問題の研究は非常に幅広く行われており，主要なテーマとして以下のものがある．

① スカイラインの効率的な計算
② ドミネイトの概念を拡張したスカイライン
③ User-Preference を考慮したスカイライン
④ 集合の傾向を把握するためのスカイライン

上記①として，D&C[1][2]，NN[4]，BBS[5]の研究がある．大まかにはインデックスを使用するアプローチ（NN, BBS），使用しないアプローチがあるが（D&C），インデックスを使用するアプローチには部分的なスカイラインを高速に求められるという特徴がある[4].

上記②として，評価基準とする次元が大きくなるとドミネイトされない点が増加する問題（次元の呪い）に対応するために k-ドミネイトの概念を導入した研究[10]がある（ドミネイトの概念は 3.1 節で説明する）．また，数値属性でない定性属性（色、サイズ、カテゴリ等）のドミネイトも扱うことが可能な m-ドミネイトの概念も存在し，定性属性，数値属性双方を考慮したスカイラインが計算できる[6].

上記③として，任意の次元のスカイラインを高速に計算できる SkyCube[11][12]や，ユーザ毎の定性属性上の嗜好を反映したスカイラインを高速に計算する研究[7]がある．

上記④として，スカイラインの厚さを考慮した K-Skyband[8]がある．スカイラインを取り去って再度スカイラインを求めたとき，取り去ったスカイラインを「K ＝ 0 のスカイバンド」と言い，再度求めたスカイラインと取り去ったスカイラインとを併せて「K＝１のスカイバンド」と言う．スカイバンドにより，その集合がどの程度の厚みを持っているのかの傾向を大まかに把握できる．また，[9]では ϕ-ドミネイトやα-ドミネイトの概念，および複数属性を考慮した集合の歪みや各点の間の密度を表現できる (ϕ,α)-quantile という指標を導入し，ある集合の傾向や特徴を可視化している．

本研究は，ある指標を使用して集合の傾向や特徴を明らかにするのではなく，他の集合と比較することによって明らかにする．

3. リッジ問合せ

本節ではリッジ問題を定義し，リッジポイントを抽出するためのアルゴリズムを提案する．なお，既存研究で提案された Naïve アルゴリズム，Skyline アルゴリズムの説明は割愛する．

3.1 リッジの定義

d 次元空間 S が与えられたとき，空間 S 中のデータセット $D = \{p_1, p_2, \ldots, p_n\}$ について，p_i の第 k 座標値を p_{ik} と表記する．

定義 1 ドミネイト(dominate)

$1 \le \forall k \le d$ に対して $p_{ik} \le p_{jk}$ であり，かつ $1 \le \exists t \le d$ に対して $p_{it} < p_{jt}$ であれば，ポイント p_i は $p_j (\neq p_i)$ をドミネイトしていると言う．ポイントp_i が $p_j (\neq p_i)$ をドミネイトする場合，$p_i \prec p_j$ と記述する．

定義 2 スカイライン（Skyline）

d 次元空間 S において，$p_j (\neq p_i)$ にドミネイトされない p_i が存在する場合，p_i は“空間 S でのスカイライン・ポイント”であり，そのような p_i の集合は“空間 S でのスカイラ

イン”であると言う．

定義 3 リッジ(Ridge)

d 次元空間 S 中の二つのデータセット P,Q が与えられ，$\exists p_i \in P$ が $\forall q_j \in Q$ に対して $q_j \prec p_i$ となるとき，p_i は，“空間 S での Q に対する P のリッジ・ポイント”であり，そのような p_i の集合は“空間 S での Q に対する P のリッジ”であると言う．図 1 では，集合 Q に対する集合 P のリッジは{p_1, p_2, p_3, p_4 }であり，集合 Q の点{q_1, q_3, q_5, q_7}を結んだ線（いわゆるスカイライン）よりも左もしくは下の斜線領域に存在している．

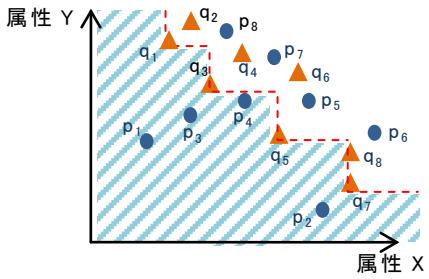


図 1 リッジの例

3.2 提案アルゴリズム

本研究では集合 P の比較対象となる集合 Q の単次元の最優位点（単次最優点），及び仮想最優点に着目した Zoom-in アルゴリズムを提案する．単次最優点とは各属性（次元）で最も優位となる点のことであり，仮想最優点とは単次最優点を組み合わせた当該集合の中で最も優位となる仮想点である．Zoom-in アルゴリズムの基本的な発想は，集合 P,Q の点を順次比較するのではなく，Q の単次最優点と仮想最優点を利用して P を段階的に絞り込み，比較する点を順次削減していくというものである．

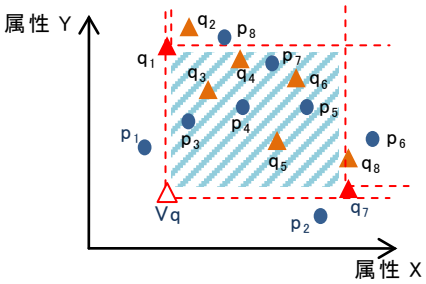


図 2 Zoom-in によるリッジ探索

手順としてはまず，集合 Q から単次最優点{q_1,q_7}を抽出して仮想最優点 Vq を作成する（図 2 参照）．集合 P と仮想最優点 Vq を比較してドミネイトされない集合 P の点{p_1,p_2}を確定リッジポイントとして比較対象から除外する．次に，仮想最優点 Vq にドミネイトされた集合 P の点（{p_1,p_2}以外の全ての点）と単次最優点{q_1,q_7}とを比較し，ドミネイトされる{p_6,p_8}を比較対象から除外する．さらに，単次最優点{q_1,q_7}と集合 Q とを比較し，ドミネイトされる{q_2,q_8}を比較対象から除外する．最後に集合 Q の比較対象の点から単次最優点{q_1,q_7}を除外し，同様の一連の処理を比較対象がなくなるまで繰り返し行う（詳細は図 3 参照）．

```
function Zoomin(P, Q)
begin
Q_b := Extract_best_points(Q)
Q_v := Create_virtual_points(Q_b)
Compare(p ∈ P, q_v ∈ Q_v) do
  if p ≯ q_v then
    R := R ∪ p
    Release(P, p)
  end
end
unless P = ∅ then
  Compare(p ∈ P, q_b ∈ Q_b) do
    if p ≻ q_b then release(P, p)
  end
  Compare(q ∈ Q, q_b ∈ Q_b) do
    if q ≻ q_b then release(Q, q)
  end
  release(Q, Q_b)
  if P ≠ ∅ && Q ≠ ∅ then
    Zoomin(P, Q)
  else
    R := R ∪ P
    return R
  end
else
  return R
end
end
```

図 3 Zoom-in アルゴリズム

集合 P と Q の分布範囲の違いが大きい場合，特にリッジが大きくなる方向に集合 P が分布している場合，効率的に比較対象となる集合 P の点を削減することができ，性能向上が期待できる．

一回目の繰返し処理にて仮想最優点 Vq が集合 P の全ての点をドミネイトできないときに計算量は最小 $\Omega(n)$になる．一方，集合 Q がマイナスの傾きの直線上に重複なしで分布し，なおかつ集合 P の全点がその直線を二等分する位置にあるときに計算量は最大 $O(n^2)$ になる．Zoom-in の計算量の詳細はその他の実験アルゴリズムの計算量とともに第 4.4 節の表 3 に記載する．

4. 実験と評価

4.1 実験対象アルゴリズム

本実験では比較対象とする集合 P,Q の相対的な位置関係・距離の違い・密度の違いが以下の各アルゴリズムの性能に与える影響を調査し，どのような場合に各々のアルゴリズムが有利となるのかを明らかにする．実験では集合 Q に対する集合 P のリッジを求めるものとする．

① Naïve
② Skyline-BNL
③ Skyline-DC
④ Zoom-in

上記①は第 3.1 節の定義と同様にリッジを抽出する手法である．上記②は集合 Q のスカイラインを求めた後にそれを利用して集合 P の点を削減して求める手法であり，スカイラインを求めるアルゴリズムには BNL(Block Nested Loop)を使用する．上記③では前述の BNL の代わりに D&C(Divide & Conquer)という，全スカイラインを求める方法では理論上最も効率的な手法を使用する．上記④は 3.2 節で提案した手法である．

4.2 実験データ

実験では一つのカテゴリ属性及び二つの数値属性を持った一様分布の人工データを使用した．数値属性のとりうる値は$[0-1)$ である．一様分布を採用したのは，集合間の距離変化の実験を行う際に集合の重なり割合を設定しやすくするためであり，結果の考察を行いやすくするためである．

また，実験シナリオ毎に 20 回人工データを生成して各アルゴリズムの処理を実行し，その平均値を実験結果とした（共通の人工データを使用して各アルゴリズムの処理を行った）．図 4 は集合 Q の位置を固定し，集合 P の位置を相対的に 0.5 の距離だけ移動させた一様分布の実験データの例である．1 列目は Left/High・Left/Middle・Left/Low，2 列目は Center/High・Center/Middle・Center/Low，3 列目は Right/High・Right/Middle・Right/Low の実験データである．Left や Low の実験ではリッジが大きくなる．

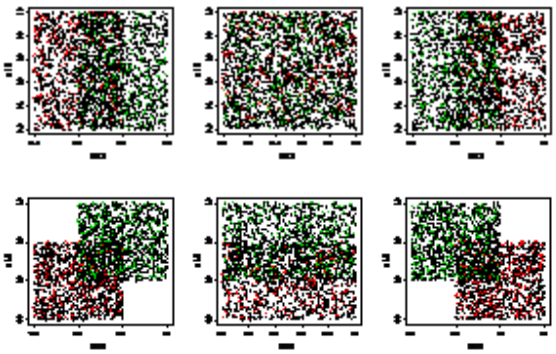


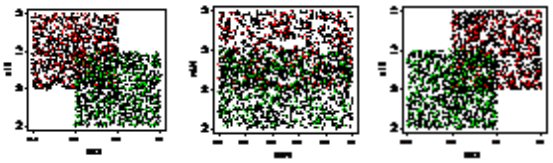
図 4 相対的な位置関係の実験データ

4.3 実験シナリオ

本研究では以下の表 1 のシナリオで実験を行った．

表 1 実験シナリオ

	位置関係	距離	密度(件数)
実験 1	Center/Middle	0.00	|P|=|Q| |P|={1000, 2000,4000, 8000,16000}
実験 2	{Left/Center /Right}{High/ Middle/Low} の全組合せ	0.50	|P|=|Q|=1000
実験 3	Left/Low	{0.00,0.25, 0.50,0.75, 1.00}	|P|=|Q|=1000
実験 4	Center/Middle	0.00	|P|=1000 |Q|={1000, 2000,4000, 8000,16000}
実験 5	Center/Middle	0.00	|P|={1000, 2000,4000, 8000,16000} |Q|=1000
実験 6	Left/Low	0.25	|P|=1000 |Q|={1000, 2000,4000, 8000,16000}
実験 7	Left/Low	0.25	|P|={1000, 2000,4000, 8000,16000} |Q|=1000
実験 8	Left/Low	0.75	|P|={1000, 2000,4000, 8000,16000} |Q|=1000



各実験の目的を表 2 に示す．

表 2 各実験シナリオの目的

	目的
実験 1	全ての実験の基準を作成する
実験 2	位置関係の影響を調査する

実験 3	距離変化の影響を調査する
実験 4	集合 Q の密度変化の影響を調査する
実験 5	集合 P の密度変化の影響を調査する
実験 6	位置関係(Left/Low)と集合 Q の密度変化とが組合わさったときの影響を調査する
実験 7	位置関係(Left/Low)と集合 P の密度変化とが組合わさったときの影響を調査する
実験 8	Zoom-in アルゴリズムの優位性を検証する

4.4 実験アルゴリズムの計算量

実験アルゴリズムの計算量を表 3 に示す.

表 3 アルゴリズムの計算量

	計算量(Order)
Naïve	$\lvert P\rvert\lvert Q\rvert$
Skyline-BNL	$\lvert Q\rvert^2 + k\lvert P\rvert\lvert Q\rvert$ $(k \ll 1)$
Skyline-DC	d = 2, 3 のとき $\lvert Q\rvert\log\lvert Q\rvert + k\lvert P\rvert\lvert Q\rvert$ $(k \ll 1)$ d = 4 以上のとき $\lvert Q\rvert(log\lvert Q\rvert)^{d-2} + k\lvert P\rvert\lvert Q\rvert$ $(k \ll 1)$
Zoom-in	$\frac{\lvert P\rvert\lvert Q\rvert}{d}(2^d - 1) + d(d+1)\sum_{j=1}^{\lvert Q\rvert/d} j$

※d は処理対象の次元(属性)の数を表す.

※k はスカイラインの大きさに左右される.

4.5 実験環境

実行プログラムは Ruby1.8.6 で実装し，以下の表 4 の環境で実験を行った.

表 4 実験環境

要素	仕様
CPU	AMD Athlon 64X2 Dual Core Processor 3800+2.00GHz
Memory	2.0GB
OS	Windows Vista Business SP1
Database	MySQL 5.0 Community Edition (Standard Parameters)

4.6 実験結果と考察

実験 1 の結果及び考察

比較対象となる集合 P,Q の分布範囲を同様（相対的な位置関係を Center/Middle）にし，密度も同様にして実験を行うと図 5 のようになった.

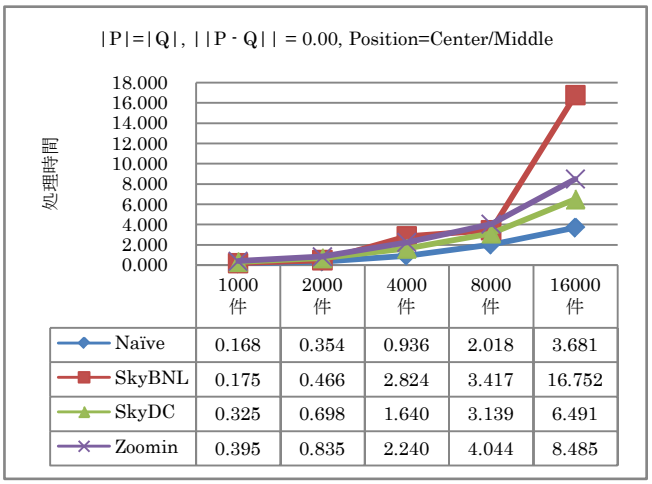


図 5 実験 1 の結果

いずれの処理件数でも Naïve が最も優位になっている. Skyline-BNL は一定件数を超過すると効率が悪化している. Skyline-BNL では比較対象の集合 Q の密度が増大するとスカイラインを求めるための比較処理が増加し，なおかつ比較処理の途中でドミネイトできない点が増加するので性能が悪化している. 集合 Q を極限まで分割してから比較処理を行う Skyline-DC は Skyline-BNL ほどの影響を受けていない. Zoom-in も安定した性能を出している.

実験 2 の結果及び考察

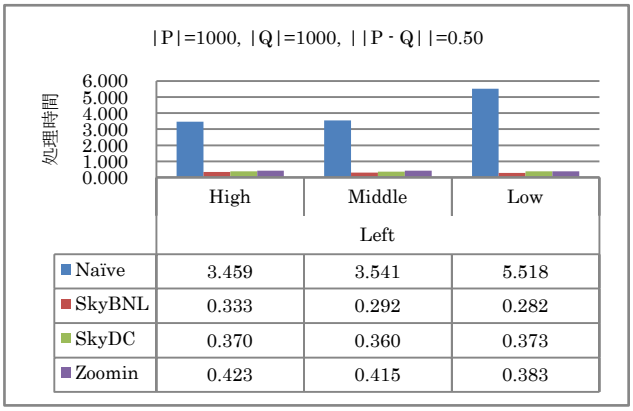


図 6 実験 2 の結果（位置=Left）

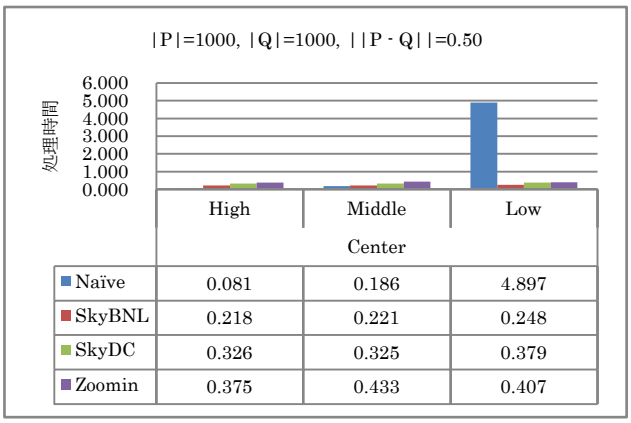


図 7 実験２の結果（位置=Center）

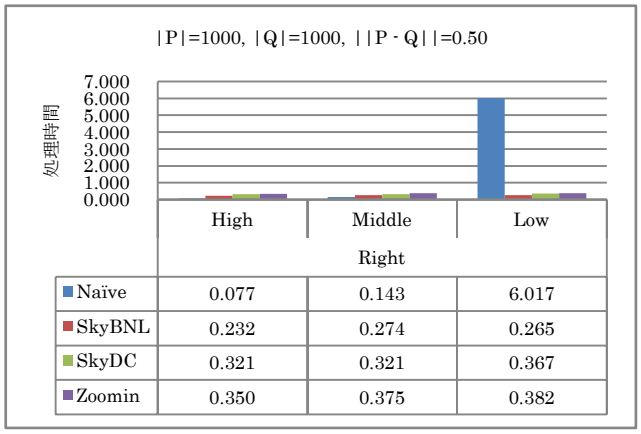


図 8 実験２の結果（位置=Right）

相対的な位置が Left の場合，High/Middle/Low いずれの場合も Naïve が際立って性能が悪化している．これはリッジが大きくなるため，比較処理の途中でドミネイトできない割合が増加するためである．位置が Center 及び Right の Low になる際にも同様の理由で Naïve は性能が悪化している．

実験３の結果及び考察

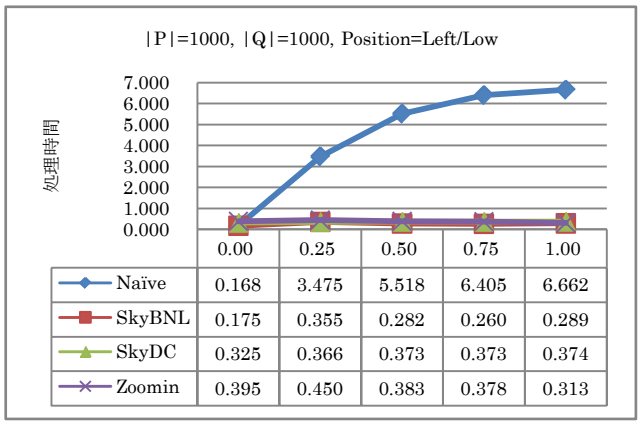


図 9 実験３の結果

集合 P をリッジが増大する方向へ移動させれば移動させるほど，Naïve の性能が影響を受けている．それは，比較処理の途中でドミネイトできない割合が高くなるためである．その他のアルゴリズムは安定しており，距離の増大による大きな影響は受けていない（Skyline-DC は若干影響を受けている）．Zoom-in はそのアルゴリズムの特性上，集合間の距離が増大するほど効果が発揮されている．

実験４の結果及び考察

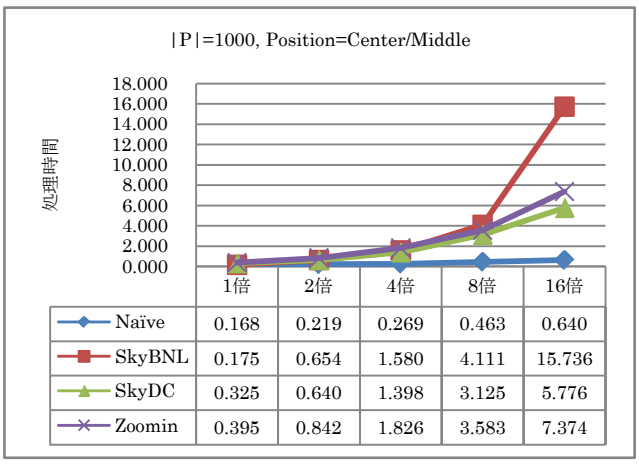


図 10 実験４の結果

集合 P に対して集合 Q の件数がある一定倍数を超過すると，Skyline-BNL はその影響を受けやすくなっている．これは集合 Q で BNL を使用してスカイラインを求める際に，処理の途中でドミネイトできない割合が高くなるためである．Skyline-DC 及び Zoom-in は密度増大の影響を受けるが，Skyline-BNL に比べて緩やかに処理時間が増加している．

実験５の結果及び考察

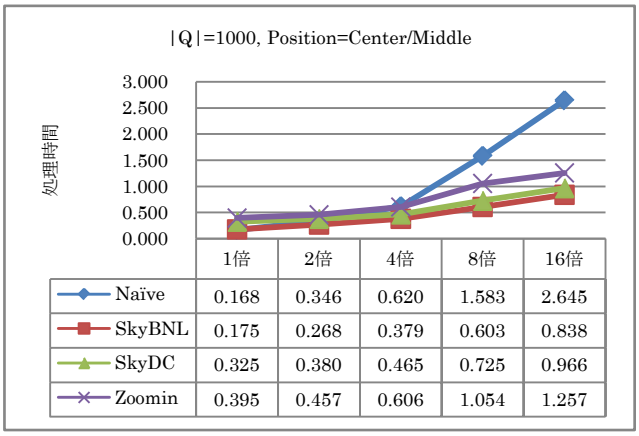


図 11 実験５の結果

集合 P の全点を比較処理するため，集合 P の件数の増加とともに Naïve が最も影響を受けやすくなっている．集合 Q の件数が増加していないため，Skyline-BNL，Skyline-DC，Zoom-in ともに安定した性能を出している．特に，集合 Q の密度が一定以下であり BNL の処理の効率が相対的に良くなるため，Skyline-BNL が高い性能を示している．

実験６の実験及び考察

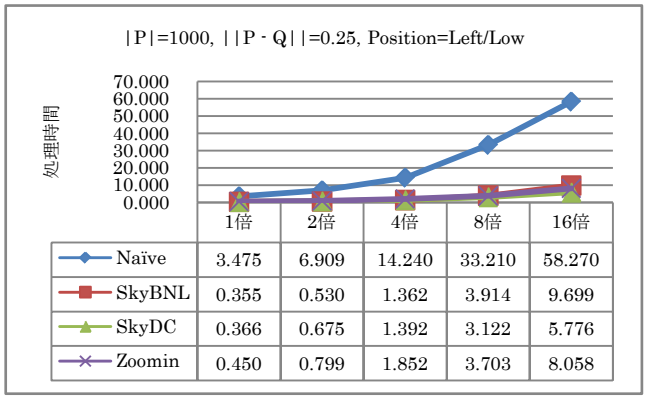


図 12 実験 6 の結果

集合 P の位置が Left/Low になってリッジが増大するため，Naïve が最も影響を受けている．集合 Q が増大してもそれを常に均等分割して一定割合で攻略するため，Skyline-DC は影響を最も受けにくくなっている．一方，Zoom-in は集合 Q を一定割合で削減していけないので，Skyline-DC に劣ると考えられる．

実験 7 の結果及び考察

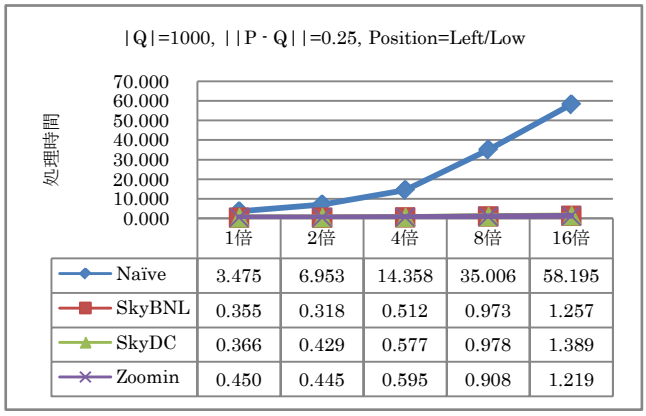


図 13 実験 7 の結果

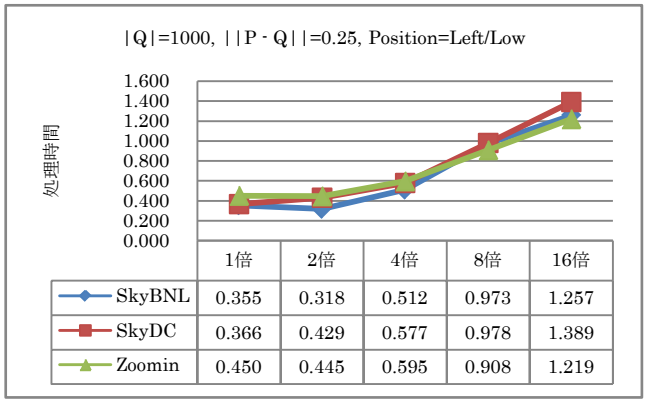


図 14 実験 7 の結果（Naïve を除外して表示）

集合 Q の密度に依存し，また集合 P,Q の重なりが小さいほど効率が良くなる Zoom-in が，集合 P の増大の影響を最も受けにくくなっている．

実験 8 の結果及び考察

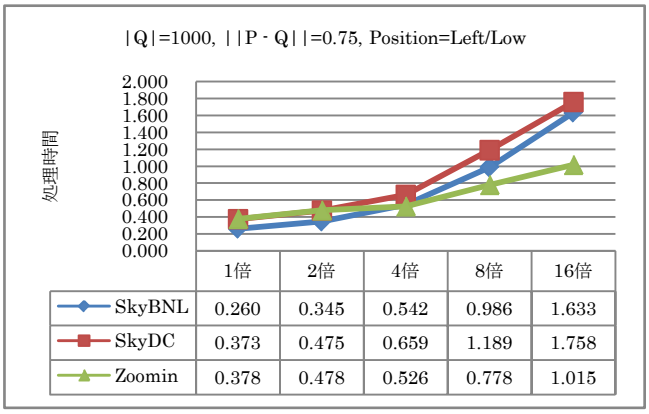


図 15 実験 8 の結果

集合 P,Q の重なりが小さくなればなるほど，なおかつリッジが大きくなる方向に集合 P が移動すればするほど Zoom-in の優位性が顕著になっている．Zoom-in は期待した効果を発揮したと言える．

実験から得られた知見をまとめると，以下の表 5 のようになる．

表 5 実験結果の考察のまとめ

	実験結果から得られた知見
Naïve	リッジが大きくなる場合，極めて性能が悪化する．また，集合 P の密度が増大したときにも性能が他のアルゴリズムと比べて悪化する．しかし，それ以外の場合はどのアルゴリズムよりも性能がよい．
SkyBNL	比較対象の集合 Q の密度が一定以下であり，集合 P の密度が一定以上である場合に最も性能がよい．しかし，Q の密度が一定以上増加すると，性能が大きく悪化する．
SkyDC	リッジが大きく集合 Q の密度が一定以上であると，最も性能がよい．
Zoomin	リッジが大きく集合 P の密度が一定以上であると，最も性能がよい．

5. むすび

本研究では，全スカイラインを求める手法の中で理論上最も効率的な D&C(Divide & Conquer)という手法

を Skyline の中に組み込んだ Skyline-DC，先行研究[2]で提案された Naïve 及び Skyline-BNL，集合間の距離が大きくなっている際に効率的にリッジを探索できる Zoom-in（本研究で提案）の四つのアルゴリズムを実験対象とし，二つの集合の密度，相対的な位置関係，距離等を変化させる網羅的な実験を行うことによって，それらのアルゴリズムの性能特性を明らかにした．Naïve は相対的位置関係や距離の変化に脆弱であり，Skyline-BNL は比較対象の集合 Q の増大に影響を受けやすいことがわかった．一方，Skyline-DC はリッジが大きく比較対象の集合 Q の密度が一定以上であるときに最も良い結果を示した．また，Zoom-in はリッジが大きく比較対象の集合 P の密度が一定以上であるときにその期待効果を最も良く発揮した．

今後の研究課題としては，三次元以上のデータでさらなる実験を行って各アルゴリズムのより詳細な特徴を明らかにすること，およびドミネイトの概念を拡張したアルゴリズムを開発すること等が考えられる．

謝辞

本研究の一部は科学研究費補助金特定領域研究（#19024006）による．

参　考　文　献

[1] S. Borzsonyi, D. Kossmann, and K. Stocker. The skyline operator. In ICDE, 2001.

[2] H. T. Kung, F. Luccio, and F. P. Preparata. On Finding the Maxima of a Set of Vectors. In Journal of the Association for Computing Machinery, Vol.22, No.4, October 1975, pp.469-476.

[3] 柿木 健, 北川 博之, 天笠 俊之, 「レコードデータに対するリッジ問合せ」, iDB フォーラム, 2008.

[4] D. Kossmann, F. Ramsak, and S. Rost. Shooting Stars in the Sky: An Online Algorithm for Skyline Queries. In VLDB, 2002.

[5] D. Papadias, Y. Tao, G. Fu, and B. Seeger. An Optimal and Progressive Algorithm for Skyline Queries. In SIGMOD, 2003.

[6] C. Chan, P. Eng, and K. Tan. Stratified Computation of Skylines with Partially-Ordered Domains. In SIGMOD, 2005.

[7] R. Wong, A. Fu, J. Pei, Y. Ho, T. Wong, and Y. Liu. Efficient Skyline Querying with Variable User Preferences on Nominal Attributes. In VLDB, 2008

[8] D. Papadias, Y. Tao, G. Fu, and B. Seeger. Progressive skyline computation in database systems. ACM Transactions on Database Systems, Vol. 30, No. 1, pp.41-82, March 2005.

[9] G. Cormode, F. Korn, S. Muthukrishnan, and D.Srivastava. Summarizing Two-Dimensional Data with Skyline-based Statistical Descriptors, 24th International Conference Scientific and Statistic Database Management, 2008.

[10] C. Chan, H. Jagadish, K. Tan, A. Tung, K. Anthony, Z. Zhang, Finding k-dominant skylines in high dimensional space. In SIGMOD, 2006.

[11] Y. Yuan, X. Lin, Q. Liu,W. Wang, J. X. Yu, and Q. Zhang. Efficient computation of the skyline cube. In VLDB, 2005.

[12] J. Lee, G. You, and S. Hwang. Telescope: Zooming to interesting skylines. In DASFAA, 2007.