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

2008－DBS－146 (46)
2008／9／23

レコードデータに対するリッジ問合せ

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 複数の定量的な属性を持つ集合の中から最良解を探索するためのスカイライン問合せが 2001 年に発表され，現在に至るまで改良がなされてきた．しかし，意思決定の場面では，代替案（集合同士）の比較を行い，優位となっているのはどの部分（タプル）なのかを識別したいときがある．本研究では，優位となるタプルを探索する問題を「リッジ問題」と定義し，複数の解法のアルゴリズムを提案して，実験により各々の性能を比較する．

キーワード リッジ問合せ，スカイライン問合せ，アルゴリズム，比較優位

Ridge Queries for Record Data

Takeshi KAKINOKI† Hiroyuki KITAGAWA†† 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 The skyline query was introduced in 2001 that can search the best answer of tuples in the data set having quantitative attributes. However, in decision-making, there are occasions where we want to recognize which tuples are comparatively dominant among multiple alternatives (sets). In this paper, we define the problem to search comparatively dominant tuples as “ridge problem”, and propose alternative algorithms to solve the problem. Then, we compare the performance of the algorithms by experiments.

Keyword Ridge Query, Skyline Query, Algorithm, Comparative Dominance

1 まえがき

複数の定量的な次元（属性）を持つデータセットの中から最良解を探索するためのスカイライン問合せが，2001 年にデータ工学の分野で発表された[1]．その後，現在に至るまで，多くの研究者によって様々な改良や拡張がなされてきた．効率的なアルゴリズムの提案[2]，ユーザ嗜好を反映した問合せ結果のランキング手法[3][4]，任意次元の効率的な逐次問合せを可能とするスカイキューブ[5]等が，その例である．

また，社会工学の分野においても多目的最適化問題として研究されており，近年，進化型アプローチの導入で発展が著しい[6]．

しかし，企業経営やマーケティング，公共政策，エンジニアリング，プロジェクトマネジメント等，実際の意思決定の場面では，あるデータセットの中での最良解を探索し，その情報だけをもとに判断を行うことはまずありえない．重要な意思決定になればなるほど，代替案の比較を厳密に行い，各代替案が他と比べて比較優位となるのはどのタプルなのかを明確に識別する必要に迫られる．

本研究では，上記のような，他の集合と比較して優位となっているタプルを探索する問題を「リッジ（山脈，分水嶺の意）問題」と定義し，リッジ問題を解くための基本アルゴリズムを提案する．

2 関連研究

スカイライン問合せでは，大量データが格納されているデータ

ベースから，任意の定量的な次元（属性）で最良となっているタプルを抽出することが可能である．本研究に関係するスカイラインの既存研究として，以下の 5 つが存在する．

① 効率的にスカイライン・ポイントを求めるアルゴリズム[2]
② 高次元でのスカイラインの評価[7][8][9]
③ 任意の次元の組合せにおけるスカイライン・ポイントの求め方（スカイキューブ）[5]
④ ユーザの嗜好を反映したスカイラインの評価[3][4]
⑤ 興味深いスカイライン近接ポイントの求め方[10]

上記①に関しては様々なアルゴリズムがあり，データを予め順序付けする SFS（Sort Filter Skyline）や R*-tree を基礎にした NN（Nearest Neighbor），BBS（Branch and Bound Skyline）などが提案されている[2]．

上記②に関しては，スカイライン問題は高次元になればなるほど，スカイライン・ポイントの数が著しく増加してしまい，大量の解が発生するという問題に対し，解決策が提示されている．代表的な手法として，“kドミナント・スカイライン”やそれに改良を加えた“k+ドミナント・スカイライン”がある[7][8][9]．

上記③に関しては，ユーザが任意の次元の組合せでスカイライン・ポイントを求めたい場合，ユーザは何度も同じような問合せを発行する必要があるので，効率が悪いという問題を議論している．例えば，”A, B, C”の次元があった場合，問合せ対象となりうる次元は” A, B, C, AB, BC, AC, ABC”の 7 つであり，問合せの度にこれらのスカイライン・ポイントを計算する必要があるので，処理に無駄が多い．

そこで，最初の問合せの際に，“スカイキューブ”と呼ばれる全ての多次元の軸を予め作成することにより，次回の問合せからは高速でスカイライン・ポイントが計算できる方法が提案されている[5]．

上記④に関しては，多次元のスカイラインを求める際，ユーザ毎に重視する次元が異なる場合の対応を議論している．あるユーザでは“A＞B＞C”の優先順位かもしれないが，別のユーザでは“B＞C＞A”かもしれない．それらのユーザの嗜好を考慮し，定量化してスカイライン・ポイントをランキング化して結果を返す手法が提案されている[3][4]．

上記⑤に関しては，通常のスカイラインでは見逃してしまうような，スカイライン・ポイントに近接するエリアから興味深いタプルを抽出する手法が提案されている[10]．

3 リッジ問合せ

本節では，新たな問題設定である「リッジ」を定義し，それに関連するポイント（タプル）をデータベースから抽出するためのアルゴリズムを提案する．

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 $p_{ik} \preccurlyeq p_{jk}$ であり，かつ $1 \le \exists t \le d$ に対して $p_{it} \prec p_{jt}$ 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．

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．

例えば，新たなリゾート地を開発するために不動産の物件を探しており，表 1 のような物件リレーション，及びタプルがデータベースに格納されていると仮定する．表 1 を 2 次元にプロットしたものが図 1 である．

表 1 物件リレーション

#	ビーチまでの距離	価格
A	1.6 km	¥30M
B	1.4 km	¥50M
C	1.2 km	¥70M
D	1.5 km	¥60M
E	1.3 km	¥100M
…	…	…
N	1.4 km	¥120M

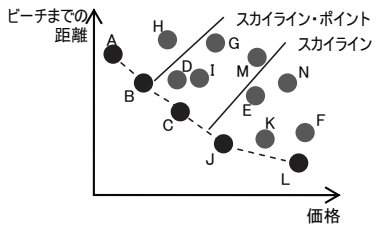


図 1 物件タプルのスカイライン

図 1 の物件“A”と“H”を比較すると，“A”はビーチまでの距離

において"H"よりも近く，有利である．また，価格においても"A"は"H"よりも安い．つまり，合理的な判断を下すならば，必然的に"H"よりも"A"が選択されることになる．定義 1 に基づくと，このような状態のことを，2 つの次元「ビーチまでの距離」及び「価格」において，"A"は"H"をドミネイトする，もしくは"H"は"A"にドミネイトされる，と表現する．

また，定義 2 に基づくと，2 つの次元「ビーチまでの距離」及び「価格」におけるスカイラインは，{A, B, C, J, L}のタプル集合である．

定義 3 リッジ(Ridge)

d 次元空間 S 中の 2 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 $p_i \succ q_j$ 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．

例えば，表 2 のリレーション及びタプルがあったと仮定し，表 2を二次元にプロットしたものが図 2である．図 2では，"沖縄に対する宮崎のリッジ"は{A, B, C, D}であり，"宮崎に対する沖縄のリッジ"は{J, K, L}である．

表 2 物件リレーション：地域表示

#	ビーチまでの距離	価格	地域
A	1.6 km	¥30M	宮崎
B	1.4 km	¥50M	宮崎
C	1.2 km	¥70M	宮崎
D	1.5 km	¥60M	宮崎
E	1.3 km	¥100M	宮崎
…	…	…	
N	1.4 km	¥120M	沖縄

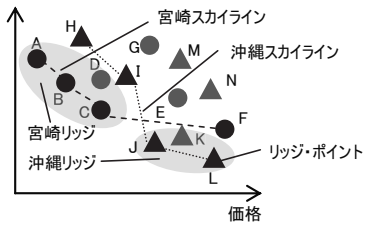


図 2 物件タプルのリッジ

本論文では，タプル集合 P, Q について，Q に対する P，もしくは P に対する Q のリッジを求める問題を考える．

3.2 アルゴリズム基本戦略

タプル集合 P, Q における，Q に対する P のリッジを求めるアルゴリズムの基本戦略として，2つの方式が考えられる．

1つは，タプル集合 P, Q を比較し合い，ドミネイトされない P のタプルを積み上げていく方式である．

もう1つは，ドミネイトすべきかどうかの基準(Q のスカイライン等)を予め求めておき，その基準をもとに差引しながら計算する方式である．

3.3 提案アルゴリズム

本研究では，3.2 節の基本戦略に従い，表 3 のような基本アルゴリズムを考案した．計算量も併記する．

表 3 基本アルゴリズムと計算量

	積上方式	差引方式
基本アルゴリズム	"Naive"	"Skyline"
	$\|P\|\|Q\|$	$\|Q\|^2 + k\|P\|\|Q\|$ $(k \ll 1, \|P\| \gtrsim \|Q\|)$

```
01: function RidgeNaive(P, Q)
02: begin
03: M1 := P , M2 := Q
04: // Compare M1 with M2 & Judge the dominance
05: foreach p ∈ M1 do begin
06:   foreach q ∈ M2 do begin
07: // Eliminate p from M1 in Memory
08:     if p ≻ q then release(M1, p), 1_break
09:   end
10: end
11: return M1
12: end
```

図 3 Naive アルゴリズム

```
01: function RidgeSkyline(P, Q)
02: begin
03: M1 := P , M2 := Q
04: // Calculating the Skyline of M2
05: MSKY := SKY(M2)
06: // Compare M1 with MSKY & Judge the dominance
07: foreach p ∈ M1 do begin
08:   foreach q ∈ MSKY do begin
09: // Eliminate p from M1 in Memory
10:     if p ≻ q then release(M1 , p), 1_break
11:   end
12: end
13: return M1
14: end
```

図 4 Skyline アルゴリズム

図 3 の Naive アルゴリズムは，リッジの定義をアルゴリズムに直接変換したものであり，タプル集合”P”の中でタプル集合”Q”にドミネイトされないタプルを抽出する．

図 4 の Skyline アルゴリズムでは，タプル集合”Q”のスカイライン・ポイントを求めた上で，タプル集合”P”のタプルと比較し，ドミネイトされない”P”のタプルを抽出する．本アルゴリズムは一般的な Skyline 処理の BNL アルゴリズムを拡張したものにしている．違いは，一般的な Skyline の BNL アルゴリズムは限られたメモリを想定し，ディスクとメモリでデータをやりとりしながら処理する方式であるが，本方式は全てのデータをメモリ内で処理する．

Skyline アルゴリズムは，”Q”が“P”と同様程度の大きさか、もしくは小さい場合に効果を発揮すると予想できる．上記の2つのアルゴリズムよりも効率的なものが存在すると容易に考えられるが，本論文では2つの方式の違いを際立たせるため，単純なアルゴリズムを採用し，4 章で実験を行った．

3.4 問合せ方法

図 5 のように，リレーショナル・データベース（RDB）の問合せ言語（SQL）の末尾に，比較対象とするタプル集合を RIDGE-OF-OVER 句で指定し，対象属性を WITH 句に記述する形式とする．

RIDGE-OF 句には複雑な条件式（WHERE 句に指定可能な条件）を記述することができ，複数のタプル集合の比較，及びリッジの抽出を行うことが可能である．

```
SELECT-FROM-WHERE…
ORDER BY-GROUP BY…
RIDGE OF <condition> OVER <condition>…
WITH <column_name> [MAX|MIN] …
```

図 5 リッジの問合せ方法

3.1 節の図 2 における”沖縄に対する宮崎のリッジ（宮崎リッジ）”を求めるには，図 6 のように記述する．

```
SELECT *
FROM 物件
RIDGE OF 地域＝‘宮崎’OVER 地域=’沖縄’
WITH ビーチまでの距離 MIN, 価格 MIN
```

図 6 リッジ問合せの具体例

4 実験と評価

4.1 実験内容

比較対象とするタプル集合“P,Q”の大きさの違い、及びタプルの分布範囲の違い（分布の重なり具合）が，各アルゴリズムの性能に与える影響を調査し，どのような場合にそれぞれ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“Q”に対する“P”のリッジを求めるものとする．

実験には，1つのカテゴリ属性及び 2 つの数値属性を持った，標準正規分布の人工データを使用した．また，集合の大きさの違いについては，“P”のデータ件数と比較して，“Q”が{2，4，8，16，32}倍の大きさとなるようにし，実験を行った．さらに，分布範囲の違いについては，“P”の分布範囲を“Q”から全体的に{(0.2, −0.2), (0.5, −0.5), (0.8, −0.8), (1.0, −1.0), (1.5, −1.5)}ずらし，性能を比較するという実験も行った（データ例は図 7 参照）．

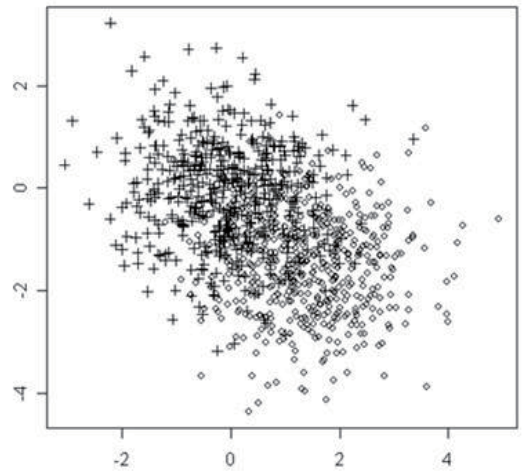


図 7 分布範囲を(1.5,−1.5)ずらしたデータの例

各実験とも 20 回ずつ人工データを生成して実施し，その平均値を実験結果とした．

4.2 実験環境

実行プログラムは 3.3 節で提示したアルゴリズムとほぼ同様に Ruby1.8.6 で実装し，以下の環境で実験を行った．

・CPU： AMD Athlon 64X2 Dual Core Processor 3800+ 2.00GHz

・Memory： 2.0GB

・OS： Windows Vista Business SP1

・Database： MySQL 5.0 Community Edition （標準パラメタ）

4.3 実験結果と考察

比較基礎となる実験結果

比較対象となる集合“P,Q”の大きさ，及び分布範囲を同じにしてNaïveアルゴリズム，及びSkylineアルゴリズムの性能を検証すると，図 8 のようになった．

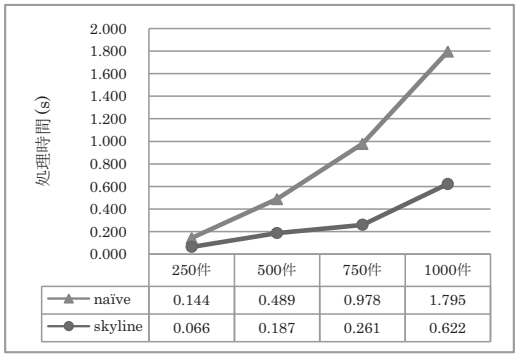


図 8 比較基礎となる実験結果

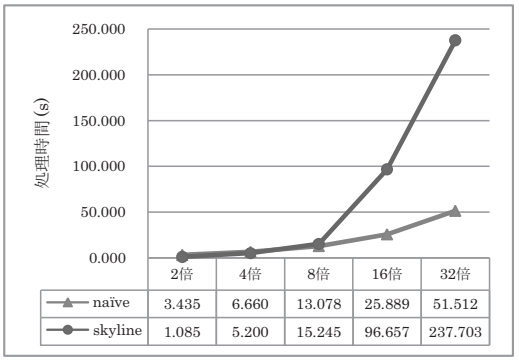


図 10 集合の大きさの違いによる影響(|P| =1000)

図 8 より，“Q”のスカイラインを求めてから“P”を差し引いてリッジを計算する Skyline アルゴリズムの方が，総じて有利であると言える．データ件数の増加に対しても，処理時間の増加傾向が低く抑えられている．

集合の大きさの違いによる影響

“P”のデータ件数と比較して，“Q”が{2, 4, 8, 16, 32}倍の大きさになると，性能は図 9，図 10 のようになった．“P”が 500 件だとすると“Q”は 500×N 件となるので，倍数が上がる毎に処理するデータ件数も増加し，必然的に処理時間は長くなる．

当実験で確認したいのは，“Q”が“P”よりどの程度大きいならば，Naïve アルゴリズムが Skyline アルゴリズムよりも有利になるのかということである．

図 9，図 10 より，Naïve アルゴリズムと Skyline アルゴリズムの性能が逆転するのは，“Q”が“P”より 8 倍弱以上の場合であることが確認できる．

また，処理するデータ件数が増加するほど，8 倍以上の際の Skyline アルゴリズムの性能が劣化する傾向が目立つ．処理データが 500 件の際は，“Q”の大きさが 16 倍，32 倍の際の処理時間の差が 4 倍弱であったのに対し，1000 件の際は 5 倍弱に拡大しているのがわかる．

分布範囲の違いによる影響

“P”の分布範囲を“Q”から全体的に{(0.2, −0.2), (0.5, −0.5), (0.8, −0.8), (1.0, −1.0), (1.5, −1.5)}ずらし，性能を比較すると図 11，図 12 のようになった．X 軸の mv02～mv15 はそれぞれ，座標をずらした値を示している．

当実験で確認したいのは，分布範囲の違いが性能に影響するか否かである．

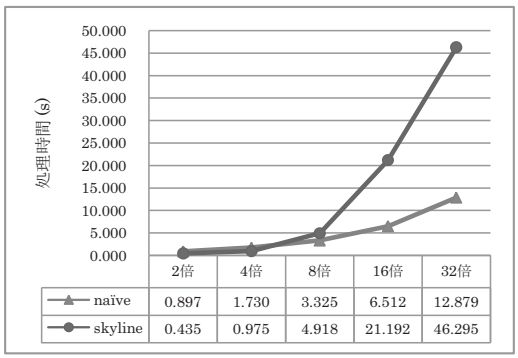


図 9 集合の大きさの違いによる影響(|P| =500)

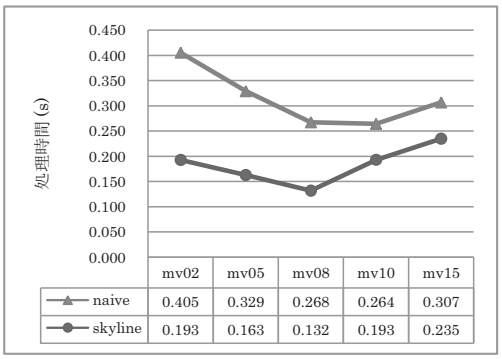
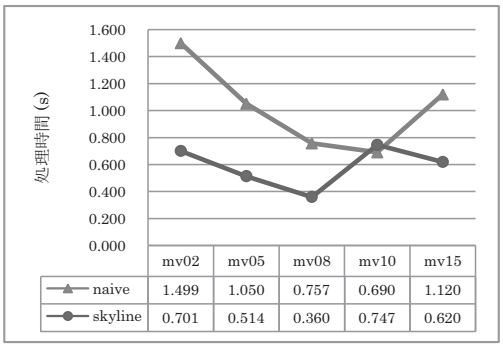


図 11 分布範囲の違いによる影響(|P| =500)



	mv02	mv05	mv08	mv10	mv15
naive	1.499	1.050	0.757	0.690	1.120
skyline	0.701	0.514	0.360	0.747	0.620

図 12 分布範囲の違いによる影響(|P| =1000)

図 11, 図 12 から, 分布範囲をずらすことにより, Naïve アルゴリズム, Skyline アルゴリズム双方に影響があると言える.

両アルゴリズムとも概ね“mv02～mv08”にかけて性能が改善しているのは, “P”が X 軸方向にプラスされたことにより, 局所的にドミネイトされやすい“P”のタプルが増加し, 処理時間が短縮していると考えられる.

しかし, 一方で, 全体としては“P”が X 軸方向にプラス, Y 軸方向にマイナスされているので, ドミネイトされないタプル(つまり, リッジ・ポイント)は増加し, ずらす度合をさらに大きくすると, ドミネイトされずに“Q”の最後のタプルまで比較処理されやすくなり, 処理時間が再び増加に転じていると考えられる.

図 12 の X 軸“mv10”において Skyline アルゴリズムが Naïve アルゴリズムよりも処理時間がかかっているのは, 相対的にデータのばらつきが大きかったためである.

図 12 の Skyline アルゴリズムの分散の平均値(“mv02～mv15”)は“0.182”であるが, Naïve アルゴリズムは“0.050”であった. “mv10”では Skyline アルゴリズムは”0.245“, Naïve アルゴリズムは”0.010“の分散となっており, データのばらつきに大きな違いがあることがわかる.

以上を総括すると, 本実験より結論付けられるのは, 以下の内容である.

・比較対象の“P,Q”の集合が同じ大きさ, 同じ分布になっていれば, 全体的に Skyline アルゴリズムの方が有利である.

・集合“Q”の大きさが集合“P”の大きさの一定以上であれば, Naïve アルゴリズムの方が有利である.

・集合の分布範囲の違いは処理性能に影響し, Skyline アルゴリズムは相対的にばらつきが大きい.

上記内容により, 当該機能を DBMS に加える際には, 事前処理として集合の大きさや分布範囲に関する統計値の取得を行い, 問合せ最適化を行う必要があると言える.

5 むすび

既存のスカイライン問題を発展させてリッジ問題を定義し, 当該問題を解決するためのアルゴリズム基本戦略, 及び基本アルゴリズムを提案した. また, 実験により, 集合の大きさの違い, 及び分布範囲の違いが各アルゴリズムの性能に影響を与えることを確認した. 今後の課題としては, リッジ問題固有の効率的なアルゴリズムを考案すること, 及び様々なデータ分布を用いて性能の違いを確認することが考えられる.

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.

文献

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

[2] D. Papadias, Y. Tao, G. Fu, and B. Seeger. An optimal and progessive algorithm for skyline queries. In SIGMOD, 2003.

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

[4] J. Lee, G. You, I. Sohn, S. Hwang, K. Ko, Z. Lee. Supporting personalized Top-k Skyline Queries using partial compressed Skycube. In WIDM, 2007.

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

[6] 中山 弘隆, ほか. 多目的最適化と工学設計―しなやかシステム工学アプローチ. 現代図書社, 2008.

[7] C.-Y. Chan, H. Jagadish, K.-L. Tan, A. K. Tung, and Z. Zhang. Finding k-dominant skyline in high dimensional space. In SIGMOD, 2006.

[8] C.-Y. Chan, H. Jagadish, K. Tan, A. K. Tung, and Z. Zhang. On high dimensional skylines. In EDBT, 2006.

[9] 仲 諒子, 遠山 元道, 高次元における k+Skyline 探索, D4-3, In DEWS, 2008.

[10] W. Jin, J. Han, and M. Ester. Mining thick skylines over large databases. In PKKD, 2004.