スカイラインの近傍探索を可能とする 拡張スカイライン演算の実装と評価

吉武 亮 † 宮崎 純† 山本豪志朗† 加藤 博一†

† 奈良先端科学技術大学院大学 〒 630-0192 奈良県生駒市高山町 8916 番地の 5
E-mail: †naist.ryo@gmail.com, ††{miyazaki,goshiro,kato}@is.naist.jp

あらまし 本稿ではスカイラインを帯状に拡張し，スカイライン周辺のデータを抽出する手法の実現方法とその評価について述べる．スカイライン演算では支配関係にのみ基づいて決定されるスカイライン集合以外のデータを抽出することが出来なかったため，例えば，情報推薦などの場面でユーザの嗜好を反映した推薦などが行うことが出来なかった．従来手法にはスカイライン点の付近のみの近傍探索手法やスカイライン演算の支配関係の緩和などによる拡張手法があった．しかし，スカイライン点周囲のみしか探索出来なかったり，データ間の距離を定量的に扱うことができない問題があった．そこで，本研究ではスカイラインを帯状に拡張し，スカイライン周辺のデータを網羅的に探索することができる拡張手法の提案と評価を行う．

キーワード スカイライン演算，問い合わせ処理

Implementation and Evaluation of an Extended Skyline Operator Searching Neighborhood of Skyline

Ryo YOSHITAKE†, Jun MIYAZAKI†, Goshiro YAMAMOTO†, and Hirokazu KATO†

† Nara Institute Science and Technology
8916-59 Takayama Town, Ikoma City, Nara Prefecture,
630-0192 Japan
E-mail: †naist.ryo@gmail.com, ††{miyazaki,goshiro,kato}@is.naist.jp

1. 序 論

近年, 計算機の処理技術の発達や記録装置の大容量化, 安価化により, データベースは大規模化してきた．データベースは様々なデータを格納することができるようになり，今後もその規模は増大の一途をたどると考えられる．大規模なデータベースからユーザにとって有用と判断されるような優秀なデータを抽出することは情報推薦や，意思決定の場で非常に重要である．データベースの大部分を占める瑣末なデータをフィルタリングせずにすべてのデータを比較，考慮することは時間的, 計算機的なコストが大きい．そのため，データベース内に含まれる優秀なデータを抽出することは重要な研究対象となっている．

Stephan Börzsönyi らは大規模データベースから優秀なデータを抽出することができるスカイライン演算 [1] を提案した．スカイライン演算は大規模なデータベースにおいてもデータの比較を行うだけで他のデータよりも優秀なデータを高速に抽出することができる．スカイライン演算で優秀なデータとして抽出されるには, データが持つすべての要素において他のデータより優れているか同等の値を持ち，少なくともひとつの次元では他のデータより優れていなければならない．優秀なデータは複数の劣ったデータを支配することができ，それらを代表することができる．抽出されたデータは互いに優れた点もあれば劣っている点もあるが，互いに支配することができないため，少なくともデータベースに格納されている抽出されなかった他のデータよりも優れていることが保証されている．

例えば，宿泊費と最寄り駅からの距離が格納されている二次元のホテルのデータベース表 1 においてスカイライン演算を実行した場合を考える．この時，最寄り駅からの距離と宿泊費が小さければ小さいほど優秀なデータであると定義する．この条件では宿泊費と最寄り駅からの距離も小さいホテルはそれより大きいホテルを支配し，代表することができる．ホテル 1 はホテル 3 と比べて宿泊費が小さく，最寄り駅からの距離は同等の値を持つため，ホテル 1 はホテル 3 を支配することができる．同じようにホテル 2 は宿泊費と最寄り駅からの距離が小さいた

表 1　ホテルデータベース

ホテル ID	宿泊費	距離
1	7500	350
2	5000	500
3	7550	350
4	5500	600
⋮	⋮	⋮

めホテル 4 を支配することができる．

ホテルデータベースに格納されているデータを宿泊費 (Price) 軸と最寄り駅からの距離 (Distance) 軸の二次元平面上にプロットし，スカイライン演算を行った時の様子を図 1 に示す．図 1 に表れている折れ線がスカイラインであり，スカイライン上にある点をスカイライン点とよぶ．スカイライン上にあるスカイライン点の集合をスカイライン集合と呼び，スカイライン集合を抽出結果として出力する．

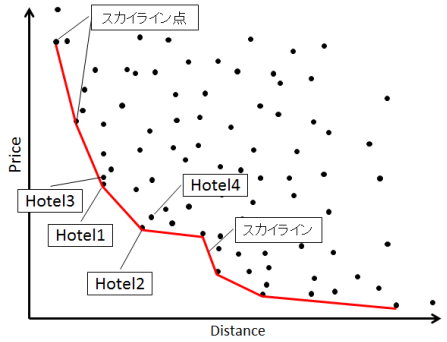

図 1　2 次元におけるスカイライン演算

スカイライン集合に含まれるには他のいかなるデータに対し支配されてはいけないため，表 1 のホテル 1 とホテル 3 の支配関係のようにホテル 3 は宿泊費がわずかに劣っているだけにも関わらず，ホテル 1 に支配されてしまい抽出されることはない．スカイラインからの距離がわずかに離れているだけにも関わらず, このようなデータが抽出されないのはデータ間の距離などを定量的に考慮しておらず，お互いのデータの優劣にのみに注目しているからである．このため, スカイライン演算は柔軟性を欠いていた．スカイライン演算によって抽出されたデータのみを情報推薦や, 意思決定の場で用いることは必ずしも有効であるとは限らないため, 実用的な問題に応用することが困難であった．

スカイライン演算は優秀なデータを抽出することができる優れたアルゴリズムであるが柔軟性がないために，現実的な問題に応用することが困難である．情報推薦の場面においてスカイライン演算を行うと個人の趣味嗜好を反映することができなかったり，意思決定の場では状況に応じて抽出結果を変化させることができなかった．本研究ではスカイライン付近やスカイライン点付近には他のデータに比べて十分に優れているデータが存在している点に着目し, スカイライン付近のデータを近傍探索を行うことができる索引構造を用いてスカイライン集合に付加して抽出する手法を提案する．

2. 関 連 研 究

これまでにスカイラインやスカイライン点の近傍に存在するデータを抽出する研究がなされてきた．スカイラインやスカイライン点の近傍のデータを抽出するための手法にはデータ間の距離に着目したものと支配関係に着目したものがある．また, 高次元においてスカイライン集合に含まれるデータ数が爆発的に増加してしまうため, そのデータ数を削減する研究が行われてきた．これは高次元では一つのスカイライン点が支配できるデータ数が減少してしまうためである．増加の傾向は, 特に負の相関を持つデータ分布ではスカイライン集合に含まれるデータが非常に多くなることが知られている．

また, スカイライン演算の研究とは直接関係しないが, 近傍領域を探索するために用いたデータ構造について関連技術として述べる．

2.1　距離に着目して近傍探索を行う研究

スカイライン付近にも優れたデータが存在していることに注目し, 距離を元に近傍探索を行う Wen Jin らの Thick Skyline [2] はスカイライン点の周辺に球状の近傍領域を設定し, その領域に含まれるデータを付加して問い合わせる手法を示した．先に挙げたホテルデータベースで Thick Skyline を行なった時の概念図を図 2 に示す．球状の近傍領域に含まれるデータはスカイライン集合に付加して抽出されるため, スカイライン点にわずかに劣ったデータも抽出される．

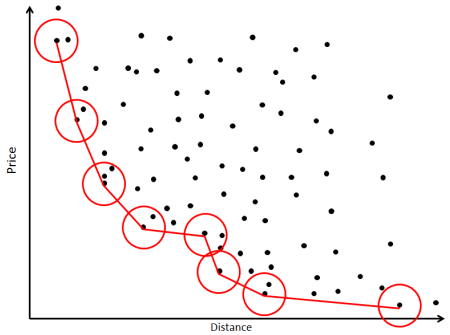

図 2　Thick Skyline

この手法ではスカイライン点との優劣を距離によって定量的に評価することができるため, スカイライン点から大きく離れたデータが抽出されることはない．しかし, スカイライン点同士が非常に離れたデータ分布では, スカイラインの中央付近に存在するデータを抽出することができない．

2.2　支配関係に着目して近傍探索を行う研究

Domitris Papadias らの k-skyband [3] ではスカイライン演算における支配条件の緩和に着目してスカイライン周辺のデータを問い合わせる手法を提案した．従来のスカイライン演算においてはスカイライン集合に含まれるには他のいかなるデータに支配されていてはいけないことが条件であった．しかし, Domitris Papadias らの手法では k 個のデータに支配されていてもスカイライン集合になることができる．先に上げたホテルデータベースで k-skyband を行なった時の概念図を図 3 に

示す.

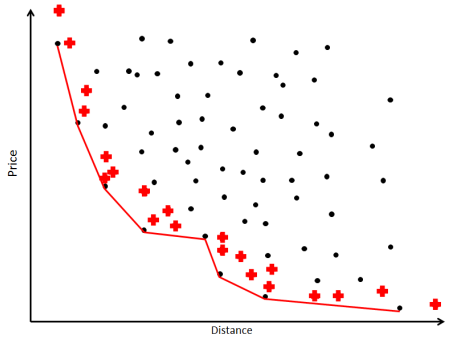

図 3　skyband

図 3 中に + で表されるデータがスカイライン集合に付加して抽出されるデータである．この手法によりスカイライン点周辺のみでなくスカイライン全体の近傍に存在するデータを問い合わせることができるようになった．しかし, この手法では支配関係にのみ着目しているため, スカイライン集合との優劣を定量的に評価できない．そのため, 支配関係の緩和ではデータ間の優劣を定量的に扱うことができないため，パラメータ k の値によっては非常に多くのデータが問い合わされたりスカイラインから遠く離れたデータも問い合わされることもある．

2.3　スカイライン集合を削減する研究

高次元になるに連れてひとつのスカイライン点が支配できるデータが少なくなり，結果的に抽出されるスカイライン集合が低次元のものと比べて大きなものになってしまう．特に負の相関のデータを持つデータベースではこの傾向は顕著である．そこで，Chee-Yong Chan らによる k-Dominant Skylines [4] ではスカイライン集合の数を減らすことを目的として提案された．k-Dominant Skylines では被支配条件を緩和し，k 個の次元で他のデータより優れていたら他のデータを支配することができると定義しデータベース全体から問い合わされるスカイライン集合に含まれるデータの削減を可能とした．

2.4　関 連 技 術

ここではスカイラインの近傍のデータを抽出する研究とは直接関係しないが, スカイライン周辺の近傍領域を探索するために用いた関連する技術について述べる．

2.4.1　SR-Tree

SR-Tree [10] とは高次元データに対する最近傍探索を高速に行うための索引構造である．これらの索引構造は多次元情報の索引のために用いられ, 一般的には範囲問い合わせを行うために用いられる．代表的な索引構造である R-Tree [7] では矩形で,SS-Tree [9] では球形の問い合わせ領域を持つのに対し,SR-Tree では矩形と球形を併用して範囲問い合わせを行うことができる．本来,SR-Tree は R*-Tree [8] を最近傍探索のために改良されたものであるが, 本研究ではスカイライン周辺に存在するデータを抽出するために利用した．

3.　提案する近傍領域と領域内のデータ抽出

これまでの研究でスカイライン周辺のデータを距離条件や支配条件によって抽出する提案がなされてきた．しかしながら, スカイライン点の分布によってはデータを抽出することができない領域が発生したり, データの優劣を定量的に評価することができなかった．

これらの問題を解決できる近傍領域は以下の要件を満たす必要がある．

（1）　スカイライン周辺のデータを抽出できる．

（2）　抽出されるデータの優劣を定量的に評価でき, 抽出されるデータの数や性質を制御できる．

（3）　重視する次元に対して重み付けを行うことができる．

(1) に関してはスカイライン点付近だけでなく, スカイライン周辺にもスカイラインと比較して十分に優れていると思われるデータが存在する可能性があるため, スカイライン全体の周辺を探索することが必要である．

(2) に関しては k-Skyband のように支配関係に着目して近傍領域を探索すると, 抽出されるデータがスカイライン点と比べてどれほど離れいているか, 知ることができないため, 距離に基づいて近傍領域を設定する．また, 支配関係に基づいて抽出されるデータの制御はパラメータの値に依存し, 抽出されるデータ数の制御が難しい．そのため, 抽出されるデータ数の制御をしやすい距離に基づいた近傍領域であることが望ましい．

(3) に関しては情報推薦や意思決定の支援などにおいてはすべての次元に対して同じ尺度で近傍探索を行うことは効果的であるとは考えにくい．嗜好や状況に応じて近傍領域の大きさを変化させることで異なる次元にたいして異なる尺度で近傍探索を行うことが必要である．

これらの要件を満たす近傍領域は距離に基づき，スカイライン全体を網羅するような近傍領域である．さらに，近傍領域は各次元に対して重み付け可能で各次元に対して異なる尺度で探索可能でなければならない．このような近傍領域はスカイライン上を楕円体が大きさを変えながら移動した時の軌跡によって表現される．その近傍領域をホテルデータベースのスカイラインに設定した時の様子を図 4 に示す．

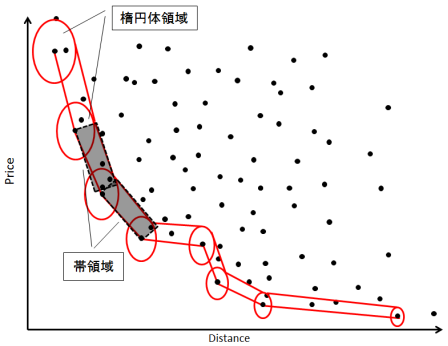

図 4　提案する近傍領域

この提案する近傍領域では, スカイライン点周辺だけではなく, スカイライン全体を近傍領域として指定することができる．楕円体の大きさを調整することで距離に基づいて近傍領域を設定することができ, 抽出されるデータの優劣を定量的に評価することができる．距離に基づいた近傍領域のため, 抽出される

データの数や, 性質を制御することができる．楕円体が大きさを変化させながらスカイライン上を移動した軌跡が近傍領域となるため, 楕円体の大きさを調整することである次元に対して重み付けを行うことができる．また, 楕円体の各次元に対する軸の長さを調整することによって各次元に対する近傍領域の広がりを変化させることができる．

なお，この近傍領域は楕円体領域と帯領域の二つに分類される．楕円体領域はスカイライン点を中心とする楕円体によって決定され，帯領域は前後の楕円体領域の共通外接線によって決定される．

3.1 全体の流れ

楕円体がスカイライン上を大きさを変化させながら移動した軌跡を近傍領域とし, その近傍領域に含まれるデータを探索するため, 以下の段階の手順を行う．

（1） 索引構造の構築
（2） スカイライン演算
（3） スカイライン点の順序付け
（4） 楕円体領域の設定, 探索
（5） 帯領域の設定, 探索
（6） 重複する近傍データの削除

これらの手順によって抽出されたデータをスカイライン集合に付加して出力する．

3.2 近傍探索のための索引構築

スカイライン付近のデータを抽出するためにスカイラインに用いたデータを用いて索引構造を構築し, 範囲問い合わせを行えるよう前処理を行う．範囲問い合わせとはある領域に含まれるデータを抽出する処理である．スカイライン付近の領域を指定して範囲問い合わせを行うことでスカイライン付近のデータを探索する．一般的に範囲問い合わせを行うために R-tree などが用いられるが, 多次元空間中の範囲問合せができる索引であれば特に限定しない．

3.3 スカイライン演算

近傍領域はスカイラインに基づいて決定されるため，まず，スカイライン演算によりスカイラインを求める．このとき，スカイライン演算を求めるための支配関係の設定は値が小さければ小さいほど優秀であるとしてスカイライン演算を行うことですべての支配関係に対応することができる．例えば，高ければ高いほど優秀であるような次元は，その次元のすべての値に対して大きな定数値から減算することで小さければ小さいほど優秀であるという支配関係に変換することができる．

小さければ小さいほど優秀であるという支配関係の優秀さの定義では探索すべき領域はすべての次元においてスカイライン点より大きな値，もしくは同等の値を持つ領域にのみ限定することができる．もしこの領域以外にデータが存在していれば，そのデータはスカイライン点として抽出されるため，図 5 のように探索領域を限定することができる．

3.4 スカイライン点の順序付け

スカイライン集合に含まれるスカイライン点の順序付けはスカイライン点同士の距離に基づいて行われる．スカイライン点 S_1 から次のスカイライン点 S_2 はスカイライン集合に含まれるデータのうち, 最もスカイライン点 S_1 に近い距離にあるスカイライン点が選ばれる．スカイライン点 S_2 の次のスカイライン点 S_3 は同様にスカイライン点 S_2 に最も近い距離にあるスカイライン点が選ばれる．最初のスカイライン点 S_1 はある任意の次元に対して最小, あるいは最大の値を持つものを選ぶ．

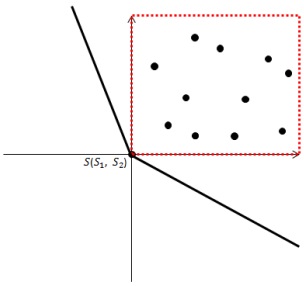

図 5　探索すべき領域

3.5 近傍領域の探索

3.5.1 楕円体領域の探索

次にスカイライン点の周囲に楕円体の近傍領域, 楕円体領域を設定する．この楕円体領域により, スカイライン点周辺のデータを抽出する．楕円体領域は大きさを自由に設定することができ, 順序付けされたスカイライン集合の順序に従って楕円体領域を自由に増加させたり, 減少させたりすることができる．また, 楕円体領域は真球の球領域であってもよく, 球領域は楕円体の各軸の長さが同じである特殊形として扱うことができる．

楕円体領域では, 楕円体を包含する矩形を設定し, 範囲問い合わせを行う． スカイライン点,$S(S_1, S_2, S_3, \cdots, S_n)$ において, 楕円体の各次元に対する楕円体の軸の長さ, $(Axis_1, Axis_2, Axis_3, \cdots, Axis_n)$ によって, 範囲問い合わせの最小値は $(S_1, S_2, S_3, \cdots, S_n)$, 範囲問い合わせの最大値は $(S_1 + Axis_1, S_2 + Axis_2, S_3 + Axis_3, \cdots, S_n + Axis_n)$ と設定することができる．二次元における楕円体領域の探索範囲を図 6 に示す．この最大値, 最小値によって決定された矩形領域で範

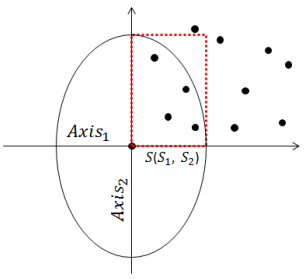

図 6　二次元における楕円体領域の探索範囲

囲問い合わせを行ない, 楕円体の中にあるデータのみを残す．楕円体の中にあるかそうでないかの判定は, 楕円体の式に代入し正負で判定する．中心が $S(S_1, S_2, S_3, \cdots, S_n)$ であり, 各次元に対する楕円体の軸の長さ $(Axis_1, Axis_2, Axis_3, \cdots, Axis_n)$ によって楕円体の式は以下のように決定される．

$$\frac{(x_1 - S_1)}{Axis_1^2} + \frac{(x_2 - S_2)}{Axis_2^2} + \cdots + \frac{(x_n - S_n)}{Axis_n^2} = 1 \qquad (1)$$

式 (1) より, 楕円体領域の中に範囲問い合わせによって抽出され

たデータが存在するかどうかを判断する関数 $D_{eli}(x_1, \cdots, x_n)$ を式 (2) に定義する．

$$D_{eli}(x_1, \cdots, x_n) = \frac{(x_1 - S_1)}{Axis_1^2} + \cdots + \frac{(x_n - S_n)}{Axis_n^2} - 1 \quad (2)$$

関数 $D_{eli}(x_1, \cdots, x_n)$ の値が 0 以下であるならば楕円体の内部にデータが存在すると判定する．

次に楕円体領域が球体であった時を考える．球体による球領域は各軸の長さが同じ楕円体の特殊形として考えることができる．球領域の中心をスカイライン点 $S(S_1, S_2, S_3, \cdots, S_n)$ とし，半径 r で範囲問い合わせを行う時を考える．SR-Tree のように球形の範囲問い合わせを行うことができる索引構造では, 中心座標であるスカイライン点 $S(S_1, S_2, S_3, \cdots, S_n)$ と半径 S のみで範囲問い合わせを行う領域が決定する．R-Tree のように矩形の範囲問い合わせしか行うことができない索引構造では, 中心座標である $(S_1, S_2, S_3, \cdots, S_n)$ を矩形の最小値とし, $(S_1 + r, S_2 + r, S_3 + r, \cdots, S_n + r)$ を矩形の最大値として与えられる矩形領域で問い合わせを行ない, 抽出されたデータに対してスカイライン点との距離を計算して距離が r 以下のデータのみを残す．

3.5.2 帯領域の探索

帯領域は隣接する楕円体領域による円錐とスカイライン点が存在する平面との断面によって求めることができる．この円錐は隣接する楕円体領域を内接する円錐である．ここでは計算を簡単にするため，楕円体領域を球と仮定して説明する．

円錐の方程式を得るには軸となる軸ベクトルと母線と軸の角度 θ, 頂点座標の 3 つの情報が必要である．これらは高次元においても条件は同じであるため, 何次元であっても円錐の方程式を求めることができる．まず, 円錐の方程式を決定するためには軸となるベクトルと頂点の座標を求める．図 7 では三次元におけるスカイラインにおける円錐の例であるが, n 次元における $S_1(S_{1_1}, S_{1_2}, ...S_{1_n})$ と $S_2(S_{2_1}, S_{2_2}, ...S_{2_n})$ の円領域が内接する円錐 E を考える．

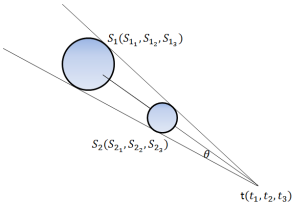

図 7　前後の楕円体領域による円錐

この円錐 E の頂点 t はスカイライン点 S_1, S_2 からなる直線の延長線上にある．スカイライン点 S_1, S_2 はそれぞれ半径 r_1 , r_2 を持ち，r_2 は r_1 よりも δ だけ変化しているとする．円錐 E の軸ベクトルと母線のなす角 θ による $\cos\theta$ は前後のスカイラインの変化による差 δ とスカイライン点同士の距離 $|S_1S_2|$ を用いて図 8 に表されているように距離による三角形の相似を用いて与えられ, 式 (3) によって決定される．

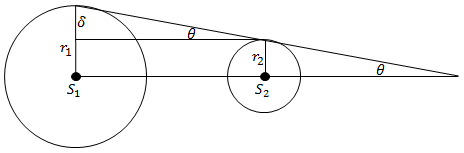

図 8　前後の楕円体領域による円錐

$$\cos\theta = \frac{|S_1S_2| - \delta^2}{|S_1S_2|} \quad (3)$$

円錐 E 頂点は S_1 から S_2 に向かうベクトル S_v を r_1/δ 倍した点に存在する．円錐 E の頂点を $t(t_1, t_2, t_3, ...t_n)$ とすると円錐 E の式は式 (4) によって求められる．

$$\begin{aligned} &\cos^2\theta((S_{1_1} - t_1)^2 + \cdots + (S_{1_n} - t_n)^2) \\ &\qquad ((x_1 - t_1)^2 + \cdots + (x_n - t_n)^2) \\ &= ((S_{1_1} - t_1)(x_1 - t_1) + \cdots + (S_{1_n} - t_n)(x_n - t_n))^2 \end{aligned} \quad (4)$$

次にある面で円錐の切断した時の楕円体を計算するために円錐 E を原点に平行移動させる．平行移動後の円錐 E_0 の頂点座標を $t_0 = (t_1 - S_{1_1}, t_2 - S_{1_2}, ..., t_n - S_{1_n}) = (t_{0_1}, t_{0_2}, ..., t_{0_n})$ とおくと，円錐 E_0 は

$$\begin{aligned} &\cos^2\theta(t_{0_1}^2 + ... + t_{0_n}^2)((x_1 - t_{0_1})^2 + ... + (x_n - t_{0_n})^2) \\ &\qquad = (t_{0_1}(x_1 - t_{0_1}) + ... + t_{0_n}(x_n - t_{0_n}))^2 \end{aligned} \quad (5)$$

と表すことができる．円錐の方程式 (5) を整理すると式 (6) により二次形式によって表現する．

$$\begin{aligned} &D_1x_1^2 + D_2x_2^2 + ... + D_nx_n^2 + \\ &E_{12}x_1x_2 + E_{23}x_2x_3 + ... + E_{(n-1)n}x_{n-1}x_n + \\ &\qquad F_1x_1 + F_2x_2 + ... + F_nx_n + f = 0 \end{aligned} \quad (6)$$

式 (6) より行列 A, 行列 B を以下のようにおくと

$$A = \begin{pmatrix} D_1 & E_{11}/2 & \cdots & E_{1n}/2 \\ E_{11}/2 & D_2 & & \vdots \\ \vdots & & \ddots & \vdots \\ E_{1n}/2 & \cdots & \cdots & D_n \end{pmatrix}$$

$$B = \begin{pmatrix} F_1 & F_2 & \cdots & F_n \end{pmatrix}$$

式 (6) は式 (7) によって表現しなおすことができる．

$$x^tAx + Bx + f = 0 \quad (7)$$

なお,$x = (x_1, x_2, \cdots, x_n)^t$ である．

行列 A は対称行列なので直行行列 P で以下のように対角化できる．

$$P^{-1}AP = \begin{pmatrix} H_1 & & & 0 \\ & H_2 & & \\ & & \ddots & \\ 0 & & & H_n \end{pmatrix}$$

一次の項に対しても同様の作業を行なう．

$$\begin{pmatrix} F_1 & F_2 & \cdots & F_n \end{pmatrix} P = \begin{pmatrix} M_1 & M_2 & \cdots & M_n \end{pmatrix}$$

以上の行列計算によって図 10 で傾いた座標系における楕円体は式 (8) によって与えられる

$$H_1X_1^2 + M_1X_1 + \cdots + H_nX_n^2 + M_nX_n + f = 0 \qquad (8)$$

式 (8) の $X_1, X_2, \cdots, X_n$ に対してそれぞれ平方完成を行うことで断面に現れる楕円体の式へ変換することができる．

$$\frac{(X_1 + M_1/2H_1)^2}{(M_1^2/4H_1 + M_2^2/4H_2 + \cdots + M_n^2/4H_n + f)/H_1} + \frac{(X_2 + M_2/2H_2)^2}{(M_1^2/4H_1 + M_2^2/4H_2 + \cdots + M_n^2/4H_n + f)/H_2} + \cdots + \frac{(X_n + M_n/2H_n)^2}{(M_1^2/4H_1 + M_2^2/4H_2 + \cdots + M_n^2/4H_n + f)/H_n} = 1 \quad (9)$$

式 (9) の楕円体を $X_1, X_2, ..., X_n$ 座標系で包含する矩形を元々のデータ空間 $x_1, x_2, ..., x_n$ 座標系で包含する矩形でレンジクエリを作成する．

図 9 は三次元における円錐と x_1x_2 平面の断面の様子を示す．図 10 は x_1x_2 平面との断面に現れた楕円体の例を示す．この楕円体によって切断した平面, もしくは空間上での最大値最小値が決定する．切断した平面, 空間以外の最大値は図 10 に表されている点 p を用いて円錐 E の頂点 t からの点 p の比を用いて求めることができる．点 p は断面に現れた楕円体の軸のひとつを選ぶことによって決定される．

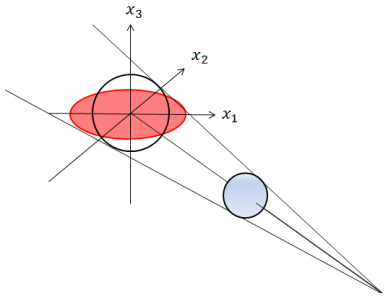

図 9　三次元の円錐と x_1x_2 平面における断面

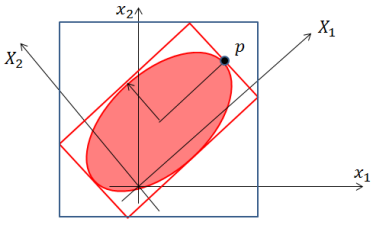

図 10　x_1x_2 平面に現れる楕円体

最後に決定した範囲問い合わせ領域の最大値, 最小値に S_1 の各次元の成分を足すことで元々のスカイライン点付近における範囲問い合わせを行うことができる．範囲問い合わせによって抽出されたデータは円錐の外にあるものも含まれているので抽出されたデータが円錐の内にあるかどうかを判断する．式 (6) を元に関数 $F_{corn}(x_1, \cdots, x_n)$ を導く．関数 $F_{corn}(x_1, \cdots, x_n)$ は式 (10) で表され, 関数 $F_{corn}(x_1, \cdots, x_n)$ を用いてデータが円錐の内側にあるかどうかを判断する．

$$F_{corn}(x_1, \cdots, x_n) = D_1x_1^2 + D_2x_2^2 + ... + Dnx_n^2 + E_{12}x_1x_2 + E_{23}x_2x_3 + ... + E_{(n-1)n}x_{n-1}x_n + F_1x_1 + F_2x_2 + ... + F_nx_n + f \quad (10)$$

抽出されたデータを式 (10) に表される $F_{corn}(x_1, \cdots, x_n)$ に代入し,$F_{corn}(x_1, \cdots, x_n)$ の値が 0 以下なら円錐内にデータがあると判断する．

楕円体がスカイライン上を移動した時の近傍領域を考える．楕円体による円錐は底面が円ではなく，楕円となる．そのため, データが存在する座標系 Σ_r における楕円体による範囲問い合わせの領域を座標変換行列 S によって座標変換し, 楕円体が球体となるような座標系 Σ_v で円錐による範囲問い合わせの座標を求める．座標系 Σ_v における範囲問い合わせの座標が求められたら座標変換行列 S の逆行列 S^{-1} によって座標変換し, 座標系 Σ_r における範囲問い合わせの座標に変換し, 範囲問い合わせを行ない, 楕円体による円錐の内にあるか外にあるかの判定を行う．

4. 実　　験

本章では拡張スカイライン演算の妥当性を検証するために提案システムを実装し，評価実験を行った結果について述べる．なお, 実験環境を表 2 に示す．

表 2　実 験 環 境

OS	Linux 2.6 Ubuntu 10.10
CPU	Core(TM)2 Quad 3.00GHz
RAM	4.0GB
HDD	1.0TB

4.1 実 験 準 備

実験は負の相関，無相関，正の相関を持つ 3 種類のデータベースにおいて行なった．それぞれのデータベースには 10 万個のデータが格納されており，各データは 0 から 1 までの値を持つ．また，これらのデータベースにおいて SR-Tree による索引構造の構築し，範囲問い合わせを行える環境を整えた．

4.1.1　スカイライン演算とその結果

2 から 10 次元のデータベースにおいてスカイライン点の数を求めた．スカイライン演算法は様々な手法 [5] [6] が提案されているが特に限定しない．なお，今回は Block Nested Loop(BNL) [1] でスカイライン点を求めた．その時の様子を表 3 に示す．

スカイライン点が高次元になるに連れてどのデータ分布においても増加している．これは高次元になるに連れて一つのスカイライン点が支配できるデータ数が減少してしまうためである．

4.2 提案手法の実行結果

ここでは二次元における提案手法を実行した時の様子について示す．この実験では,0 から 1 までの値を持つ負の相関 (Anti),

表 3　スカイライン集合要素数

次元	負の相関	無相関	正の相関
2	52	8	6
3	347	58	22
4	1781	276	69
5	5560	694	84
6	12529	2051	120
7	22713	4870	192
8	35886	9943	226
9	49571	15772	304
10	62787	24621	358

無相関 (Indp)，正の相関 (Corr) を持つ 1000 個の 2 次元のデータベースにおいて提案手法を行った．楕円体領域をを球領域とし，先頭のスカイライン点の球領域の最初の半径，初期近傍半径を 0.05，末尾のスカイライン点の球領域の半径，最終近傍半径を 0.001 として比例配分によって大きさを変化させながら球体がスカイライン上を移動した時の軌跡を近傍領域とした．その時のスカイラインとスカイラインの近傍領域に含まれるデータを図 11, 図 12, 図 13 に示す．図中の折れ線がスカイラインであり，丸く強調されているデータが提案手法によって抽出されたデータである．

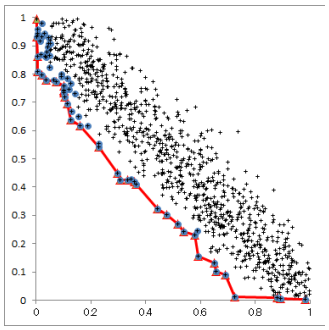

図 11　負 の 相 関

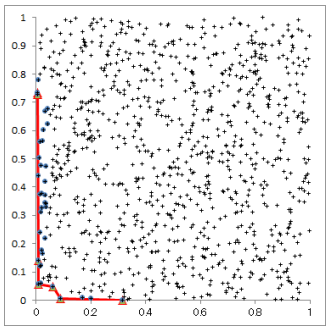

図 12　無　相　関

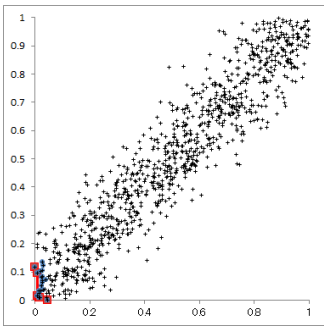

図 13　正 の 相 関

近傍に含まれるデータ数は負の相関, 無相関, 正の相関それぞれにおいて 63,36,21 であった．どのデータ分布においても先頭のスカイライン点付近が多くデータが取られており, 特に負の相関においては $k = 4$ の k-skyband で現れる $(0.0294, 0.9427)$ といったスカイライン点から離れたデータも抽出された．もし,$k = 4$ で k-skyband を行なっていたら末尾のスカイライン点付近でも多くのデータが抽出されるのに対し, 本手法ではデータは抽出されず, 横軸の値が小さい時ほどデータを多く抽出するという, 重み付けを行うことができた．

一方,Thick Skyline との比較ではスカイライン点付近の抽出されたデータの差はほとんど見られなかったが, スカイライン点とスカイライン点との中央付近に存在するデータ特に無相関におけるスカイライン点 $(0.0046, 0.7274)$ と $(0.0089, 0.1410)$ の間に存在するデータが多く抽出されており,$(0.0361, 0.4733)$, $(0.0327, 0.6680)$ といったデータ,24 個が抽出されていた．また, 末尾スカイライン点付近のデータはほとんど抽出されておらず, 横軸に対する重み付けを行うことができた．

4.2.1　楕円体による近傍領域に含まれるデータ

楕円がスカイライン上を移動した時の近傍領域によって抽出されるデータの比較を行う．今回使用する楕円体は横軸に対して縦軸に 2 倍の広がりを持つ．図 14 と図 15 の右上に近傍領域に用いた真球と楕円体を示す．実際の近傍領域に用いられた真球や楕円体とは縮尺が異なるが, 楕円の長軸と短軸の比率は実際の近傍領域と等しい．

この縦に引き伸ばされた楕円体によって抽出されるデータを図 15 に丸印で強調して示す．縦に引き伸ばされているため, 同じデータセットで行ったときと異なるデータがいくつか抽出されている．球領域の近傍領域によって抽出されたデータと異なるデータが多く存在する場所を図 15 中に示す．

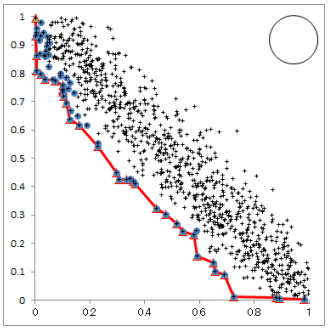

図 14　球による近傍領域

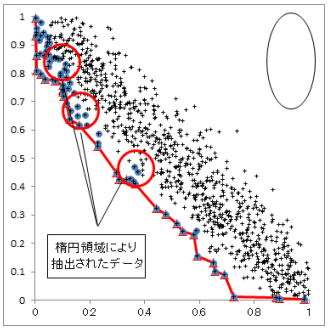

図 15　楕円体による近傍領域

楕円体領域によって抽出されたデータ数は 1000 個中 77 であり, 球領域によって抽出されたデータ数 63 を上回った．楕円体領域のみによって抽出されたデータは具体的に $(0.3438, 0.4898)$ や $(0.2309, 0.5860)$ などがあり, これらのデータを球領域の軌跡による近傍領域で抽出しようとすると先頭のスカイライン点の球領域の半径を 0.08, 末尾の球領域の半径を 0.05 に設定する必要があり, 抽出されたデータの数は 156 にのぼった

このように, 楕円体を使わずに抽出されるデータの性質を制御しようとすると, 非常に多くのデータが抽出されてしまい, スカイライン演算の優秀性を保つことが難しくなる．そのため, 各次元に対して異なる尺度で探索を行うことができる楕円体を使った近傍領域の設定は実問題への応用に効果的であると考える．

4.2.2　近傍領域探索の実行時間

提案手法の実行時間について調べた．表 4 より次元が高次元になるに連れて抽出されるスカイライン点が増加するため，実

行時間が増加していくことが確認できる．また，正の相関を持つデータベースではスカイライン点は少ないが，データが密集している領域を探索するため，実行時間が無相関のデータベースよりも実行時間が大きくなっている．

表 4　近傍探索の実行時間 (秒)

次元	負の相関	無相関	正の相関
2	0.2076	0.0328	0.2874
3	0.5992	0.1081	0.6378
4	6.4835	0.2853	2.0445
5	19.5068	1.0885	2.1328

また，近傍領域を変化させた時の実行時間と近傍領域に含まれるデータ数について調べた．三次元の無相関のデータベースにおいて BNL にてスカイライン点を求めた後，近傍領域の探索時間を測定した．近傍領域は初期近傍半径のみを変化させ，最終近傍半径を 0.001 に固定した時の実行結果を図 16 に示す．初期近傍半径が大きくなるに連れて実行時間と抽出されるデータ数が増加していることがわかる．しかし，スカイライン演算を再び行うより高速に実行することが出来ているため，ユーザの嗜好等が変化した時も近傍領域の変更によって対応することができる．また，近傍領域を変化させることで抽出されるデータ数を制御できることを示した．

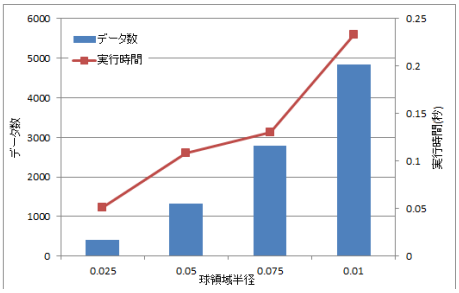

図 16　球領域の半径による抽出データ数と実行時間の変化

5. 結　　論

本稿ではスカイライン演算によって求められるスカイライン集合の周囲を R-Tree や SR-Tree などの範囲問い合わせを実行できるデータ索引を用いてスカイラインの周囲を探索する手法を提案, 評価した．本手法によって抽出されたデータはスカイライン上を楕円体が大きさを変化させながら移動した軌跡に含まれるデータであり, 楕円体の大きさの変化量を調整することでスカイライン周辺のデータを抽出することでスカイライン演算に柔軟性を持たせることができ, 情報推薦や意思決定における趣味嗜好や, 状況を考慮してデータの抽出内容を変化させることができるため, スカイライン演算を現実的な問題への対応への有用性を示した．

5.1　今後の課題と展望

本手法では高次元やデータが密集している分布では実行時間が増加してしまうことがわかった．高次元において実行時間が増加してしまうのはスカイライン点が増加してしまうためである．よって高次元のスカイライン集合に含まれるデータ数を削減する k-dominant skylines によって削減されたスカイライン点を元に近傍領域の探索を行うことで対処できると考える．

三次元以上ではスカイライン集合は三角形の集合として表現することができる．近傍領域の探索を行うためには三角形の面による柱体の近傍領域を探索する必要があるが，柱体の探索は処理的なコストが大きい．そのため，本来探索すべき領域を簡略化して三角形の各辺に対して近傍領域の探索を行うことで対処できると考える．

謝　　辞

本研究の一部は，科研費補助金基盤研究 (A)(課題番号:22240005) ならびに基盤研究 (C)(課題番号:23500121) の支援による．ここに記して謝意を表す．

文　　献

[1] Stephan Börszönyi, Donald Kossmann, Konrad Stocker ”The Skyline Operator ” In Proceeding of the 17th International Conference on Data Engineering, pp.421-430, 2001.

[2] Wen Jin, Jiawei Han, Martin Ester ”Mining Thick Skylines over Large Databases ” In Proceedings of the 8th European Conference on Principles and Practice of Knowledge Discovery in Databases, pp.255-266, 2004.

[3] Dimitris Papadias, Greg Fu, Jp Morgan Chase, Bernhard Seeger ”Progressive Skyline Computation in Database Systems ” In Proceeding of ACM Transactions on Database Systems (TODS), vol30, pp.41-82, 2005.

[4] Chee-Yong Chan, H. V. Jagadish , Kian-Lee Tan, Anthony K. H. Tung, Zhenjie Zhang ”Finding k-dominant skylines in high dimensional space” In Proceedings of the 2006 ACM SIGMOD international conference on Management of data, pp.503-514, 2006.

[5] Jan Chomicki, Parke Godfrey Jarek Gryz Dongming Liang ”Skyline with Presorting ” In Proceeding of Data Engineering 19th international Conference, pp595-604, 2003.

[6] Donald Kossamann, Frank Ramsak, Steffen Rost ”Shooting stars in the sky : an online algorithm for skyline queries ” In Proceeding of the 28th international conference on Very Large Data Bases, pp.275-286, 2002.

[7] Antonin Guttman ”R-Trees:A Dynamic Index Structure for Spatial Searching” In Proceedings of the 1984 ACM SIGMOD international conference on Management of data - SIGMOD, pp. 47-57, 1984.

[8] Beckmann. N, Kriegel. H-P, Schneider. R, Seeger. B ”The R*-tree: an efficient and robust access method for points and rectangles+” In Proceedings of the 1990 ACM SIGMOD international conference on Management of data - SIGMOD pp. 322-331, 1900.

[9] D. A. White and R. Jain, ”Similarity Indexing with the SS-tree” Proc. of the 12th Int. Conf. on Data Engineering, pp.516-523, 1996 .

[10] 片山紀生, 佐藤真一, 「SR-Tree: 高次元点データに対する最近接検索のためのインデックス構造の提案」, 電子情報通信学会論文誌 D-I, vol. J80-D-I, no. 8 (Aug. 1997) pp. 703-717