空間データベースのための凸領域分割アルゴリズムの拡張

An Extended Cell Splitting Algorithm for Spatial Databases

田中 美智子♥ 金子 邦彦♣
陸 応亮♦ 牧之内 顯文♠

Michiko TANAKA Kunihiko KANEKO
Yingliang LU Akifumi MAKINOUCHI

凸領域(cell)の超平面による分割は計算幾何学，空間データベース，線形制約データベースの分野で重要な問題である．凸領域の超平面による分割は各種の空間幾何演算（intersection，difference）の基礎となる．本論文では任意次元で動き，非有界な凸領域をも扱える凸領域の超平面による分割アルゴリズムを提案する．既存の凸領域分割アルゴリズムは k-polytope に接続する(k-1)-polytope の数が 2 個以上の場合にしか動かず，結果として有界な凸領域しか扱えない．提案アルゴリズムは，計算の過程でできる k-polytope に接続する (k-1)-polytope の数について０個，１個，２個以上の場合分けを行い，それぞれについて異なる polytope 分割判定処理，position vector 作成処理を行う．このことで非有界の凸領域を扱うことが可能となる．本論文では提案アルゴリズムの実装と，各種の評価についても報告する．

Splitting cell problem is an important problem in computational geometry, spatial database and constraint database areas. Spatial operations such as intersection and difference is based on splitting cell with hyper planes. This paper presents an algorithm to split a bounded and an unbounded spatial objects with hyperplanes in any dimension. The previous algorithm only works for a bounded objects because it assumes that all k-polytope has more than 2 (k-1)-polytopes connected with. This algorithm considers the number of (k-1)-polytopes in the evaluation process and applies different polytope splitting algorithm and position vector making algorithm This make it possible to compute an unbounded cells. This paper presents the implementation and evaluations.

1. はじめに

我々は, Hawk's Eyeという空間データベースシステムを開発してきた．このシステムでは任意の次元の空間物を扱うことができる. Hawk's Eyeでは，空間物は超平面集合によって生成されるpolytopeの集合として定義される．これは，計算幾何学に基礎を置くものである[2][3][4]．そして，このデータモデルにより，様々な次元の空間物を統一的に扱うことが可能になる．データモデルの詳細については[6]で紹介されている．

我々の目的は，有界な空間物だけでなく非有界な空間物についても, intersection や difference のような幾何演算を行う効果的なアルゴリズムを開発することである．これらの幾何演算を設計する際に，凸領域(cell)を超平面で分割するアルゴリズムは,重要な役割を担う．

本論文では，有界な凸領域と非有界な凸領域を次のように定義する．dを一つの凸領域内にある任意の2点の間の距離とする．もしd<mを満たす実数mが存在する場合は，凸領域は有界である．そうでなければ凸領域は非有界である．

提案するアルゴリズムの基礎となる部分は[7] に紹介されている．しかしそのアルゴリズムは，非有界な凸領域に対しては適用することができない．これは,そのアルゴリズムが,全てのk-polytope(k≧0)が２つ以上の(k-1)-polytopeに接続していることを前提としている為である．非有界な凸領域を構成するpolytopeの中には，この前提条件を満たさないものもある.よって非有界な凸領域を扱うために拡張を行った．

また,非有界な凸領域を分割するアルゴリズムとしては,Edelsbrunnerの超平面アレンジメント構築アルゴリズムを利用する方法もある．凸領域を構成する超平面集合と分割に使用する超平面集合で超平面アレンジメントを構築し，凸領域に含まれる部分を選びだす． しかしこの場合,大きな割合を占める凸領域に含まれない部分の分割も行う必要があるの.よって提案するアルゴリズムの方が計算量が少ない.

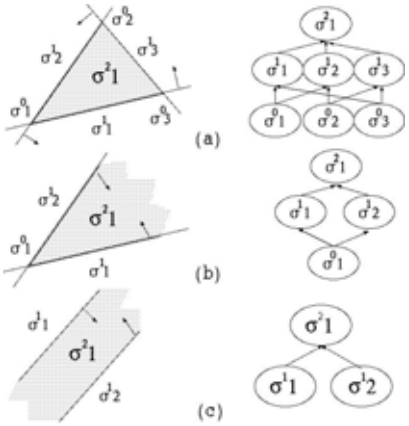

図 1 凸領域とそのgraph表現
fig 1 Cell and Cell Graph

表 1 Hasse diagram の実装
polytope を表現しているそれぞれのnodeは，この表の情報を属性として持つ
Table 1 Our implementation of the Hasse diagram.
Each node representing a polytope has its attributes

	Attributes
0-polytope	Coordinates value: (x, y, x)
	position vector
k-polytope (k>0)	position vector
	number of (k-1)-polytopes connected with the k-polytope
	number of (k+1)-polytopes connected with the k-polytope
	dimension of the k-polytope

♥ 学生会員 九州大学大学院システム情報科学府修士課程
tanaka@db.is.kyushu-u.ac.jp
♣ 正会員 九州大学大学院システム情報科学研究院
kaneko@is.kyushu-u.ac.jp
♦ 非会員 九州大学大学院システム情報科学府修士課程
riku@db.is.kyushu-u.ac.jp
♠ 正会員 九州大学大学院システム情報科学研究院
akifumi@is.kyushu-u.ac.jp

本論文では，非有界な凸領域を扱うことのできる拡張凸領域分割アルゴリズムを提案する．このアルゴリズムは１つ以下の(k-1)-polytopeに接続しているk-polytopeが存在する場合にも適用することができる．それぞれの凸領域はHasse Diagramを利用したgraphで表現することができる[8]．本論文では，凸領域を表現するgraphのそれぞれのnodeの属性にposition vectorを導入する．このposition vectorは,幾何演算を行う際に必要である．

本論文の構成は次のようになっている．2章では,Hawk's Eyeでのデータ表現を示す．3章では拡張凸領域分割アルゴリズムを説明し,４章で実験結果を示す．

2. 凸領域の表現

１つの k 次元凸領域は，0 から k 次元の polytope の集合として表現する．例えば三角形は，３つの 0-polytope と３つの 1-polytope，１つの 2-polytope により構成される．本論文において k 次元の polytope は，k-polytope と表記する．

凸領域に含まれるpolytope同士の接続関係は，graphにより表現される．このgraphはHasse Diagramと同形である[8]．我々は，その実装に独自の方法を用いた．graphのnodeの属性は表 1 に示した通りである．position vectorとは,超平面集合Hとpolytopeの位置関係を表す ＋ － ＝ を値として持ち，長さはHの要素数と等しい．ここで凸領域を構成する超平面を{φi}(1≦i≦n)とし，分割に用いる超平面をhとする．凸領域分割前はH={φi}であり,分割後はH={φi,h}である.つまり凸領域を分割するとposition vectorの長さは 1 増える． また,空間中のpolytopeは，position vectorにより識別することができる.例えば,図１(a)のσ21のposition vectorは[+++]である．position vectorは，intersection や difference といった空間幾何演算の結果を選び出す際に必要である．

図１の(a) は有界な図形，(b)と(c)は非有界な図形の例である．ここで重要なのは，有界な凸領域の graph 中の k-polytope(k≧1)は,必ず 2 つ以上の(k-1)-polytope に接続しているのに対し，非有界な凸領域の場合には(k-1)-polytope の数が 1 以下である k-polytope が存在するということである．このことが,[7]で示されているアルゴリズムが非有界な凸領域に対して適用できない原因である．

3. 凸領域分割アルゴリズム

この章においては，凸領域の分割を行なうアルゴリズムについて説明する．このアルゴリズムに対する入力は graph と凸領域を構成している超平面の集合，及び graph を分割する超平面であり,出力は超平面で分割された graph である．

3.1 polytope の分割処理

提案する拡張凸領域分割アルゴリズムは，各 k-polytope(k≧1)についてそれに接続する(k-1)-polytope の数により場合分けを行ない，それぞれの場合について異なる処理を行なう．図 2 にアルゴリズムを示す．split_2(), split_1(), split_0() はいずれも一つの polytope について，次の処理を行うアルゴリズムである．

・polytope が分割されるか否かを判定し，分割される場合はその事実を報告する．

・polytope が分割されない場合は，分割に使用した超平面に対して，polytope が ＋ , － , ＝ のいずれであるかを求める．＋ , － , ＝ は,polytope の超平面に対する位置関係を表す．

```
input:
  the cell graph G
  the set of hyperplanes construct the graph HP
  hyperplane h
  the dimension of space d
output:
  the graph split by h

1. classify all the vertices
     either ＋ or － or ＝
2. for k ← 1 to d
3.   for all k-polytope f in the graph
4.     n ← the number of (k-1)-polytope which
           f is connected with
5.     if n ≧ 2
6.       then P ← the positions of (k-1)-polytope
                  which f is connected with
7.            split_2(P)
8.       else if n=1
9.         then p ← the position of (k-1)-polytope
                    which f is connected with
10.             split_1(k,d,p,HP,h)
11.        else split_0(HP,h)
12.   if k-polytope is not split
13.     then make position vector
14.     else remove f. Make new k-polytope f+ and f-. Connect them
          with the (k+1)-polytope which f was connected with.
          For all (k-1)-polytopes which f was connected with,
          if it is ＋(－) then connect with f+(f-). Make new
          (k-1)-polytope f0. Connect it with f+ and f-.
          If k ≧ 2, connect f0 with the (k-2)-polytope which is
          ＝ and is connected with such (k-1)-polytope
          that was connected with f.
```

図 2 凸領域分割アルゴリズム

fig 2 Cell Splitting Algorithm

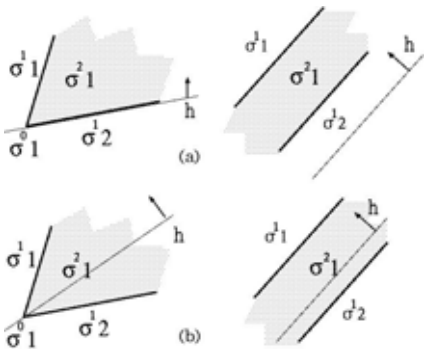

図 3 2 つ以上の (k-1)-polytope に接続する k-polytope

fig 3 k-polytope connected with more than two (k-1)-polytope

その後,polytope が分割されると判定された場合は分割により新たに作成される polytope を作成し,他の polytope との接続関係を作成するという graph の更新を行う.polytope が分割されない場合は,求めた位置関係を利用して polytope の position vector を作成する.この処理で position vector の長さは 1 長くなる．

3.2 2 つ以上の(k-1)-polytope に接続している k-polytope に対する処理

2 つ以上の(k-1)-polytope に接続している k-polytope に対しては，split_2()が呼び出される．これは[7]に示されているアルゴリズム split の step2 と同じ処理である．図３に例を示

```
input:
  the dimension of k-polytope k
  the dimension of space d
  the position of (k-1)-polytope p
  hyperplanes which contain k-polytope HP
  hyperplane by which split the k-polytope h
output:
  if k-polytope is split, return the fact.
  otherwise return ＋, －, or ＝

1.if k=d
2. then if p= ＝
3.   then if k-polytope is contained by h+
4.     then return ＋
5.     else return －
6.   else if h cross with k-polytope
7.     then return the fact that k-polytope
             is split
8.     else if k-polytope is contained by h+
9.       then return ＋
10.      else return －
11.else if k-polytope and h is parallel
12. then return p
13. else if p= ＝
14.   then if k-polytope is contained by h+
15.     then return ＋
16.     else return －
17.   else p ← the cross of h and HP
18.     if p is contained by k-polytope
19.       then return the fact that k-polytope
              is split
20.       else return p
```

図 4　アルゴリズム split_1()

fig 4　Algorithm split_1()

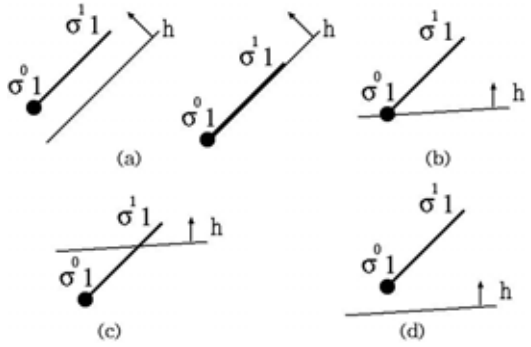

図 5　1 つの 0-polytope に接続している 1-polytope
fig 5　1-polytope connected with one 0-polytope

す.

3.3 1 つの(k-1)-polytope に接続している k-polytope に対する処理

1 つの(k-1)-polytope に接続している k-polytope の場合,split_1()が呼び出される. この内部では k-polytope は,k の値が空間次元数と同じか否かで,更に 2 つの場合に分けられる. k の値が空間次元数と同じでない場合は,k-polytope

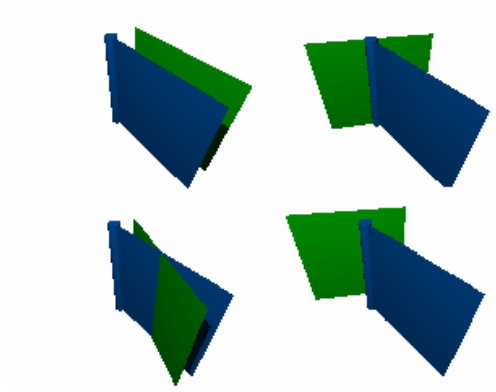

図 6　1 つの(k-1)-polytope に接続している k-polytope
fig 6　k-polytope connected with one (k-1)-polytope

```
input:
  hyperplanes which contain k-polytope HP
  hyperplane by which split a polytope h
output:
  if k-polytope is split, return the fact.
  otherwise return ＋, －, or ＝

1. if 1-polytope is parallel with h
2.   then if 1-polytope is contained by h
3.     then return ＝
4.     else if 1-polytope is contained by h+
5.       then return ＋
6.       else return －
7.   else return the fact that 1-polytope
           is split
```

図 7　アルゴリズム split_0()

fig 7　Algorithm split_0()

と超平面が並行になることがあり得る. k の値が空間次元数と同じ場合は,並行にはなり得ない.

図 5, 図 6 に例を示す. (a) k-polytope が超平面と並行な場合は, (k-1)-polytope が＋(－ ＝)なら k-polytope も＋(－ ＝)である. (b) (k-1)-polytope が超平面上にある場合は,k-polytope は ＋ か － である. 上記のいずれにも当てはまらない場合は(c) か (d) である. どちらの場合であるかを判定する為には, k-polytope を含む k 次元の領域と超平面との交差部分上にある点を計算で求める. 例えば 2-polytope が 3 次元空間にある場合は, 2-polytope を含む平面と超平面との交差部分である直線にある点を計算する. (c) もし 2-polytope が点を含んでいる場合は分割される. (d) そうでない場合は, 2-polytope が＋(－ ＝)なら 3-polytope も＋(－ ＝)である.

3.4 (k-1)-polytope に接続していない k-polytope

(k-1)-polytope に接続していない k-polytope に対しては, split_0()を呼び出す.

図 8 に例を示す. (a) 超平面と k-polytope が平行であり k-polytope が超平面上にある場合は,k-polytope は＝である. (b) 平行であるが,k-polytope が超平面上にない場合は＋か－である. (c) 超平面と k-polytope が平行でない場合

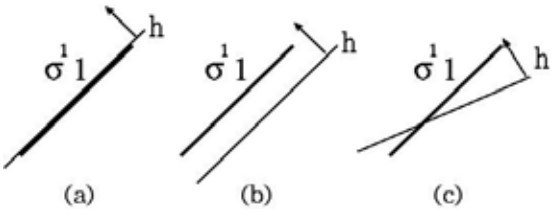

図 8 (k-1)-polytope に接続していない k-polytope
fig 8 k-polytope not connected with (k-1)-polytope

は,k-polytope は分割される.

4. 実験

我々は,提案したアルゴリズムの実装を行なった. 実験は, d次元空間中に予めd個の超平面で構成される非有界な凸領域を作成しておき, その凸領域を分割するような複数個の超平面をランダムに発生させて凸領域の分割を行った.

実験環境は, Sun Blade 100, メモリ 512MB, OS は SunOS5.0 である. 実験の結果を図 9,図 10 に示す. グラフの横軸は分割により生成された polytope の総数であり,縦軸は分割に要した時間である. 分割に使用した超平面の数は, 3 次元空間での実験では(4,8,12,16,20,24)であり, 4 次元空間での実験では(2,4,6,8,10,12,14)である. 提案したアルゴリズムでの計算時間は polytope の総数に比例すると予測していたが, この結果より確認できた.

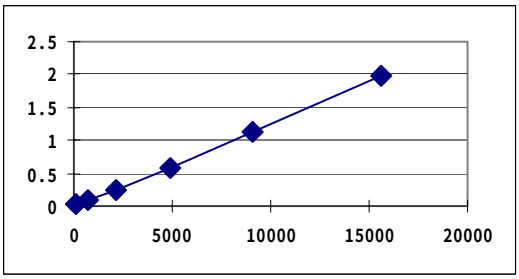

図 9 3次元空間中の非有界な凸領域を複数超平面で分割した際の時間
Fig 9 the time to split an unbounded cell with hyperplanes in 3 dimensional space

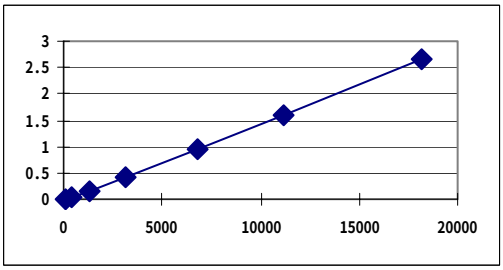

図 10 4次元空間中の非有界な凸領域を複数超平面で分割した際の時間
Fig 10 the time to split an unbounded cell with hyperplanes in 4 dimensional space

5. おわりに

我々は,[7]に示されていた凸領域分割アルゴリズムを基にして,非有界な凸領域も扱うことができるアルゴリズムを提案し,実装を行なった. 2 個以上の(k-1)-polytope に接続しているような k-polytope に対しては既存のアルゴリズムを使い, 1 個以下の(k-1)-polytope に接続しているような k-polytope に対しては, 超平面の方程式を利用するという拡張を行うことで, 非有界な凸領域を分割することが可能であった.

[謝辞]

本研究の一部は、日本学術振興会科学研究費補助金課題番号 15650017, (A)(2)16200005 による。

[文献]

[1] 今井 浩, 今井 桂子, “計算幾何学”, 共立出版株式会社 ,1994

[2] Joseph O`Rourke ,“Computational Geometry in C Second Edition”, Cambridge University press ,1998

[3] Herbert Edelsbrunner, “Algorithms in Combinatorial Geometry”, Springer-Verlag , 1987

[4] M.de Berg, M.van Kreveld, M.Overmars, O.Schwarzkopf, “Computational Geometry Algorithms and Applications”, Springer 1998

[5] Kunihiko Kaneko, Akifumi Makinouchi, “HA-face Complex Spatial Data Model for Uniform Representation of Data and Algorithms”, 2003-DBS-131 pp29, 2003

[6] Yingliang Lu, Michiko Tanaka, Kunihiko Kaneko, Akifumi Makinouchi, “Design and Implementation of a Spatial Data Model HA-face Complex and a Kernel for Spatial Database Systems Hawk’s Eye “, DEWS2004 pp4-B-05, 2004

[7] Chandrajit L. Bajaj and Valerio Pascucci, “Splitting a Complex of Convex Polytopes In Any Dimension” Proceedings of the twelfth annual symposium on Computational geometry pp88-97, 1996.

[8] TROTTER.W.T, “Combinatorcs and Partially Orderd Sets: Dimension Theory”, Johns Hopkins Series in the Mathematical Sciences. The Johns Hoplins University Press, 1992

田中 美智子 Michiko TANAKA
九州大学大学院システム情報科学府 知能システム学部門修士課程在学中. 空間データベースシステムの研究開発に従事. 日本データベース学会学生会員.

金子 邦彦 Kunihiko KANEKO
九州大学大学院システム情報科学研究院助教授. 1995 九州大学工学研究科情報工学専攻博士課程修了. 博士(工学). 空間データベースの研究に従事. 情報処理学会, 電子情報通信学会, 日本データベース学会, ACM, IEEE 会員.

陸 応亮 Yingliang LU
九州大学大学院システム情報科学府 知能システム学部門修士課程在学中. 空間データベースシステムの研究開発に従事.

牧之内 顕文 Akifumi MAKINOUCHI
九州大学大学院システム情報科学研究院教授. 1970 フランス グルノーブル大学理学部修了(Docteur-Ingeneur 授与). 生物情報データベース, XML データベース, 空間データベースの研究に従事. 日本データベース学会理事, 情報処理学会フェロー, 電子情報通信学会フェロー, ACM, IEEE 会員.