2004－DBS－134　(II)　(51)
2004／7／14

空間データベースのための拡張超平面分割アルゴリズム

田中　美智子[†]　金　子　邦　彦[††]
陸　　応　亮[†]　牧之内　顕文[††]

凸領域 (cell) の複数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. 凸領域の複数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, 凸領域の複数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,face に接続する sub-face 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 face に接続する sub face の数について, 0 個, 1 個, 2 個以上の場合分けを行った後, 超平面との交差判定を正しく行い, 新規に提案する位置ベクトル (position vector) 作成アルゴリズムを使って,face の位置ベクトルを作成する. 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.

An Extended Cell Splitting Algorithm for Spatial Database

Michiko Tanaka,[†] Kunihiko Kaneko,[††] Yingliang Lu[†]
and Akifumi Makinouchi[††]

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

1. はじめに

様々な分野において, 空間物を取り扱うことが必要とされている.Geometric information system(GIS) や computer aided design(CAD) , computer graphics(CG) , liner constraint database などは, 代表的な例である. 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, 超平面によって生成される HA-face の集合として表現される. これは, computational geometry 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. これらの幾何計算を設計する際に, 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.

この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-HA-face(1<k) が２つ以上の (k-1)-HA-face に接続していることを前提としているためである. しかし, 非有界な凸領域を構成する HA-face 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は, 拡張が必要である.

本論文では, 有界な空間物と, 非有界な空間物は以下

† 九州大学 大学院システム情報科学府
Graduate School of Information Science and Erectrical Engineering
†† 九州大学 大学院システム情報科学研究院
Faculty of Information Science and Erectrical Engineering

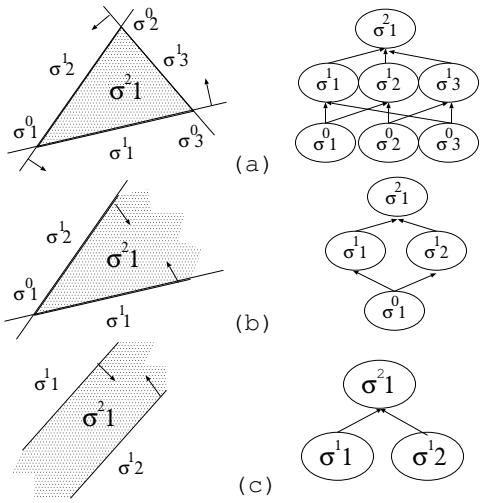

図 1 Cell and Cell Graph

のように定義する. d を一つの空間物内にある任意の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, 空間物は有界である. そうでなければ, 空間物は非有界である.

本論文では, 非有界な空間物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)-HA-face にしか接続していない k-HA-face がある場合にも, 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 の属性に位置ベクトルを導入する. この位置ベクトルは, 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 でのデータ表現と, 幾何演算の概要を示す.3 章では, 凸領域分割アルゴリズムを説明する.

2. HAWK'S EYE

この章では,Hawk's Eye での凸領域の取り扱いについて説明する.

2.1 凸領域の表現

一つの k 次元凸領域は,0 から k 次元の HA-face の集合として表現する. 例えば, 三角形は, 3 つの 0-HA-face と 3 つの 1-HA-face と 1 つの 2-HA-face により構成される. 本論文において k 次元の HA-face は,k-HA-face と表記する.

空間中の HA-face は, 位置ベクトルにより識別することができる. 位置ベクトルのそれぞれの要素は,HA-face と超平面の相対的な位置関係を表現する. 一つの超平面により, 空間は h+と h-と超平面自身の 3 つの

表 1 Hasse diagram の実装
HA-face 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

	attributes
0-HA-face	coordinates: (x, y, z) position vector
1-HA-face	position vector bounded or unbounded: boolean value number of sub-HA-faces direction: (x, y, z)
k-HA-face (k>2)	position vector bounded or unbounded: boolean value number of sub-HA-faces

領域に分割される. ある HA-face が h+(h-, 超平面自身) に含まれる場合,HA-face はその超平面に対して, position+(-,0) を持つ. 例えば,Fig.1 の $\sigma^2 1$ は,h1+と h2+,h3+に含まれているので位置ベクトルは [+++] である.

凸領域に含まれる HA-face 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,Table.1 に示した通りである. 本論文においては, sub-HA-face という言葉を, 次のように定義する.

- ある (k-1)-HA-face が k-HA-face に接続している場合,(k-1)-HA-face は, k-HA-face の sub-HA-face である.

position vector という属性は,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.position vector の長さは, k 次元の凸領域に接続している (k-1)-HA-face の数と同じである. これは, 凸領域が非有界の場合も同じである.Fig.1(a) は, 有界な三角形とその graph 表現の例である.Fig.1(b) と Fig.1(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 中には, 少なくとも 1 つの 0-HA-face が存在するのに対して, 凸領域が非有界な場合は,graph 中に 0-HA-face が一つもない場合が存在するということである. 非有界な凸領域の graph 中の HA-face の最小次元は決まっていない. そしてこのことが,[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.

2.2 空間演算

Fig.2 は,2 つの凸領域の intersection を求める方法の概要を示したものである.: (a) cell graph を構築する; (b) 他方の図形の超平面で cell graph を分割する; (c) 両方の凸領域に含まれている HA-face を選び出す

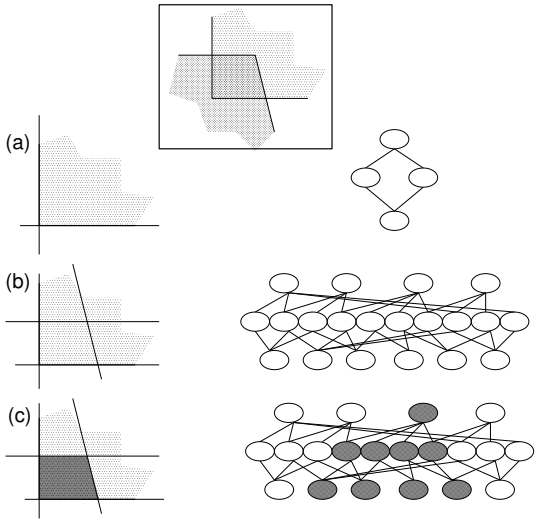

図 2 Intersection

凸領域の分割アルゴリズムは,(a) と (b) で使用される. また,(c) において, HA-face を選ぶために position vector が利用される.

2.3 非有界な凸領域の分割

すでに Edelsbruner の超平面アレンジメント構築アルゴリズムがある[3]. このアルゴリズムは, 凸領域の分割にも利用できる. まず, 凸領域の境界を構成している超平面を利用して, 超平面アレンジメントを構築する. そして分割に用いる超平面を作成した超平面アレンジメントに追加する.

しかし,Edelsbruner のアルゴリズムは, 超平面アレンジメントを構築している超平面と, 新たに追加する超平面とが交点を持つことを前提としている. よって, 例えば,Fig.1(b) のような非有界な凸領域を分割するような場合, 分割する超平面は,(b) を構成している超平面と必ず交点を持つため,Edelsbruner のアルゴリズムを使うことができる. しかし,Fig.1(c) のような凸領域を,(c) を構成している超平面と並行な超平面で分割するような場合は, Edelsbruner のアルゴリズムは適用できない. 今回提案したアルゴリズムでは, このような場合にも, 凸領域の分割を行なうことができる.

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 graph 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. Fig.3 にアルゴリズムを示す.reform() は,graph の更新を行なう処理である.

```
input:
  the graph of cell
  hyperplane h
  the dimension of space d
output:
  the graph of cell splitted by hyperplane

1. if the maximum node is bounded
2.   then split
3.   else classify all the vertices
          either '+'or'-'or'0'
4.     for all 1-HA-faces
5.       n ← the number of 0-HA-face
         which 1-HA-face is incident to
6.       if n=2
7.         then split_1_HA_face_2()
8.         else if n=1
9.           then split_1_HA_face_1()
10.          else split_1_HA_face_0()
11.      if 1-HA-face is not splitted
12.        then make position vector
13.        else reform()
14.    for k ← 2 to d
15.      for all k-HA-faces
16.        n ← the number of (k-1)-HA-face
           which k-HA-face is incident to
17.        if n>=2
18.          then split_k_HA_face_2()
19.          else if n=1
20.            then if k=d
21.              then split_k_HA_face_1_s()
22.              else split_k_HA_face_1_n()
23.            else split_k_HA_face_0()
24.        if k-HA-face is not splitted
25.          then make position vector
26.          else reform()
```

図 3 Cell Splitting Algorithm

凸領域の分割を行なう際に, まず凸領域が有界か非有界かどうかを識別する. これは, 極大元がもつ bounded か unbounded かのフラグを利用して行なう. もし, 凸領域が有界な場合は,[7] のアルゴリズム split と同様のアルゴリズムで HA-face の分割判定を行ない,position を求める. その後,HA-face が分割される場合は,reform() が呼び出される. split の入力は,graph と graph を分割する超平面であり, 出力は分割された graph である.

もし, 凸領域が非有界であった場合は,Fig.4, 6, 9, 11 に示した今回提案するアルゴリズムを使用する.HA-face が分割される場合は, その後 reform() が呼び出される.

3.1 face の分割処理

提案する凸領域 分割アルゴリズムは, 各 HA-face について, それに接続する sub HA-face 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.k-HA-face を分割する際には, それに接続している (k-1)-HA-face が 2 つ以上であるか,1 つであるか, 0 個であるかにより, 異なる処理を行なう.

アルゴリズムを,Fig.3 に示す.split_1_HA_face_2() と split_k_HA_face_2() は,sub HA-face の position を入力として受け取り, HA-face の position を返す. もし,HA-face が分割される場合は, その事実を報告する. これは,[7] のアルゴリズムと同じである.

3.2 1-HA-face に対する処理

提案するアルゴリズムでは,1-HA-face が, 以下の 3 つの場合のいずれであるかを判定する.TABLE1 に示

```
input:
  the position of 0-HA-face f
  hyperplanes which contain 1-HA-face HP
  hyperplane by which split 1-HA-face h
output:
  the position of 1-HA-face if 1-HA-face
  is splitted, return the fact

1. if 1-HA-face is parallel with h
2.  then return f
3.  else if f='0'
4.    then if 1-HA-face is contained by h+
5.      then return '+'
6.      else return '-'
7.    else p ← intersection_of_hyperplanes(HP,h)
8.      if p is contained by 1-HA-face
9.        then return the fact that 1-HA-face
              is splitted
10.       else return f
```

図 4 Algorithm split_1_HA_face_1(f,HP,h)

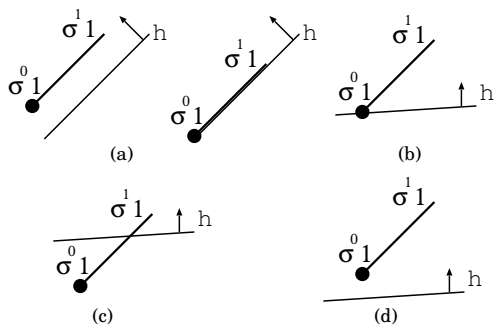

図 5 1-HA-face incident to 1 0-HA-face

したように, 各 1-HA-face は, その属性として sub-HA-face の個数を持っているので, この情報を利用する.

- 2 つの 0-HA-face に接続している (線分)
- 1 つの 0-HA-face に接続している (半直線)
- 0-HA-face に接続していない (直線)

3.2.1 1 つの 0-HA-face に接続している 1-HA-face

1-HA-face が 1 つの 0-HA-face だけに接続している場合は,Fig.4 に示したアルゴリズムが呼び出される.Fig.5 に例を示す.

超平面と 1-HA-face が並行な場合は,1-HA-face の position は 0-HA-face の position と同じである (a). それ以外の場合で,0-HA-face の position が'0' の場合は,1-HA-face の position は'+' か'-' である (b). 上記のいずれにも当てはまらない場合は, (c) か (d) である. どちらであるかを判定する為には,1-HA-face を含む直線と超平面との交点を計算する. もし, 交点が 1-HA-face に含まれるなら 1-HA-face は分割される (c). 交点が 1-HA-face に含まれない場合は,1-HA-face の position は,0-HA-face の position と同じである (d).

3.2.2 0-HA-face に接続していない 1-HA-face

1-HA-face が 0-HA-face に接続していない場合は,Fig.6 に示したアルゴリズムが呼び出される.Fig.7 に例を示す.

```
input:
  hyperplanes which contain 1-HA-face HP
  hyperplane by which split a face h
output:
  the position of 1-HA-face if 1-HA-face
  is splitted, return the fact

1. if 1-HA-face is parallel with h
2.  then if 1-HA-face is contained by h
3.    then return'0'
4.    else if 1-HA-face is contained by h+
5.      then return'+'
6.      else return'-'
7.  else return the fact that 1-HA-face
        is splitted
```

図 6 Algorithm split_1_HA_face_0(HP,h)

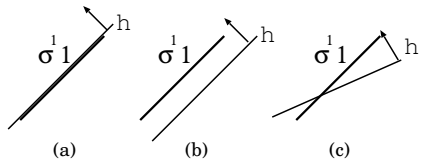

図 7 1-HA-face incident to 0 0-HA-face

超平面と 1-HA-face が並行であり, かつ 1-HA-face が超平面上にある場合は, 1-HA-face の position は'0' である (a). 超平面と 1-HA-face が並行であり, case a 以外の場合は,1-HA-face の position は'+' か'-' である (b). 超平面と 1-HA-face が並行でない場合は,1-HA-face は分割される (d).

3.3 k-HA-face(k≥2) に対する処理

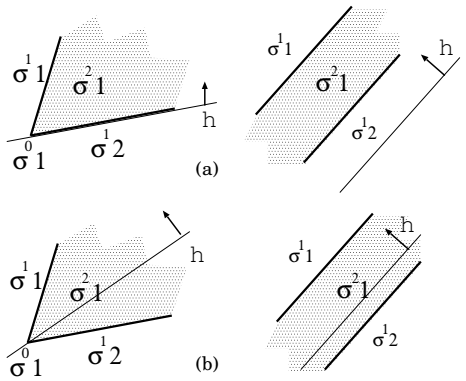

図 8 k-HA-face incident to more than 2 (k-1)-HA-face

すべての 1-HA-face について処理が終了した後,kHA-face(k≥2) について処理を行なう. 提案するアルゴリズムでは, まず k-HA-face が以下の 3 つの場合のいずれであるかを判定する. これは,k-HA-face が属性として持っている sub-HA-face の個数を利用して判定する.

- 2 つ以上の sub-HA-face に接続している
- 1 つの sub-HA-face に接続している

```
input:
  the position of (k-1)-HA-face f
  hyperplanes which contain k-HA-face HP
  hyperplane by which split the k-HA-face h
output:
  the position of k-HA-face if k-HA-face
  is splitted, return the fact

1. if f='0'
2.  then if k-HA-face is contained by h+
3.     then return '+'
4.     else return '-'
5.  else if h cross with k-HA-face
6.     then return the fact that k-HA-face
            is splitted
7.     else if k-HA-face is contained by h+
8.       then return '+'
9.       else return '-'
```

図 9 Algorithm split_k_HA_face_1_s(f,HP,h)

- sub-HA-face に接続していない

3.3.1 2 つ以上の sub-HA-face に接続する k-HA-face(k≥2)

k-HA-face が 2 つ以上の sub-HA-face に接続している場合は, 次の性質を持つ.

- position'+'('-') を持つ sub-HA-face が存在しない場合,k-HA-face の position は'-'('+') である
- 上記に当てはまらない k-HA-face は, 分割される

この性質は, 有界な凸領域と同様である.Fig.8 に例を示す.

3.3.2 1 つの sub-HA-face に接続する k-HA-face(k≥2)

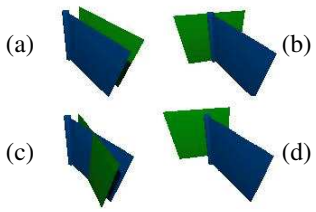

図 10 k-HA-face incident to 1 (k-1)-HA-face

この場合,k-HA-face は,'k' 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' が空間次元数と同じでない場合は,k-HA-face と超平面が並行になることがあり得る.'k' が空間次元数と同じ場合は, 並行にはなり得ない. それぞれの場合によって, 異なる処理が呼び出される.'k' が空間次元数と同じ場合は,Fig.9 に示したアルゴリズムが, そうでない場合は,Fig.11 に示したアルゴリズムが呼び出される.

Fig.10 に例を示す.k-HA-face が超平面と並行な場合は,k-HA-face の position は, sub-HA-face の position と同じである (a).(k-1)-HA-face が超平面上にある場合は,k-HA-face の position は'+か"-' である (b). 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-HA-faec 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-HA-face が 3 次元空間にある場合は,2-HA-face 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. もし 2-HA-face が点を含んでいる場合は, (c) である.2-HA-face が点を含んでいない場合は, 2-HA-face の position は 1-HA-face の position と同じである (d).

```
input:
  the position of (k-1)-HA-face f
  hyperplanes which contain k-HA-face HP
  hyperplane by which splitted the k-HA-face h
output:
  the position of k-HA-face if k-HA-face
  is splitted, return the fact

1. if k-HA-face and h is parallel
2.  then return f
3.  else if f='0'
4.     then if k-HA-face is contained by h+
5.       then return '+'
6.       else return '-'
7.     else p ← the cross of h and HP
8.       if p is contained by k-HA-face
9.         then return the fact that k-HA-face
                 is splitted
10.        else return f
```

図 11 Algorithm split_k_HA_face_1_n(f,HP,h)

3.3.3 sub-HA-face に接続していない k-HA-face(k≥2)

この場合は,0-HA-face に接続していない 1-HA-face の場合と同様の処理を行なう.

3.4 graph の更新

HA-face が分割される場合,graph の形を変更する必要がある. つまり, 分割される HA-face に対応する node を graph 中から削除し, 新しく作成される HA-face に対応する node を作り, 接続する node と正しく接続させるという処理である.

この処理は Fig.12 に示したアルゴリズムで行なう.Fig.12 中の super-HA-face は, 入力された HA-face に接続している HA-face で,HA-face より次元が 1 つ高いような HA-face のことである. また,sub-sub-HA-face とは,HA-face の sub-HA-face の sub-HA-face のことである.

4. 性能テスト

我々は, 提案したアルゴリズムの実装を行なった. 今回実験を行なった環境は,Sun Sparc Ultra10 ワークステーション, メモリ 760MB, OS は Sun0S 5.9 である.

実験の結果は,Fig.13,14 に示したとおりである. この結果より, 凸領域の分割に要する時間は,graph 中の HA-face の数にほぼ比例していることが確認できた.

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

```
input :
  HA-face to split  n
  the dimension of n  d
  hyperplane by which split n  h
  the graph of cell
output :
  the graph  HA-face is splitted

1.   make n+ which has '+' for h
     and the dimension is d
2.   make n- which has '-' for h
     and the dimension is d
3.   for all super-HA-face n_sup of n
4.     connect n and n_sup
5.   for all sub-HA-face n_sub of n
6.     if n_sub is '+' for h
7.       then connect n+ and n_sub
8.       else connect n- and n_sub
9.   make n0 which has '0' for h and
     the dimension is d-1
10.  connect n+ and n0
11.  connect n- and n0
12.  for all sub-sub-HA-face n_sub2 of n
13.    if n_sub2 has '0' for h
14.      then connect n0 and n_sub2
```

図 12 Algorithm reform()

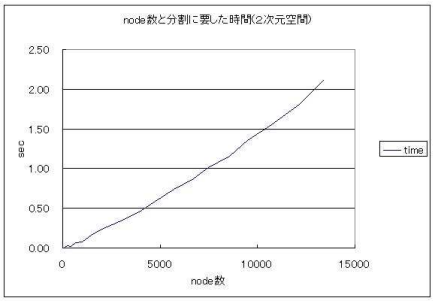

図 13 2 次元空間中の半平面を, ランダムに発生させた超平面で分割した. グラフは,k-HA-face($0 \leq k \leq 2$) の総数と, 分割に要した時間の関係を示したものである.

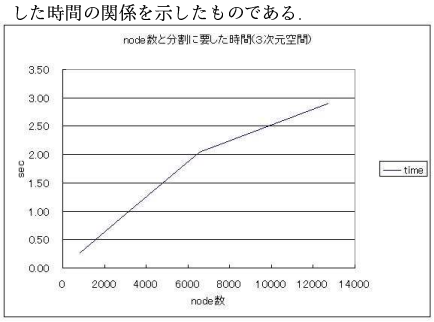

図 14 3 次元空間中の半平面を, ランダムに発生させた超平面で分割した. グラフは,k-HA-face($0 \leq k \leq 3$) の総数と, 分割に要した時間の関係を示したものである.

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. 非有界な凸領域を分割する問題は, それぞれの HA-face の sub-HA-face の数により, 処理を分けることで可能であった. また, 位置ベクトルを利用することで, 有界な凸領域と同様の方法で, 非有界な凸領域を扱うことができた.

参 考 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, 2003
6) Yingliang Lu, Michiko Tanaka, Kunihiko Kaneko, Akifumi Makinouchi, *The Uniform Representation of Algorithms for Spatial Database Systems* , DEWS2004 , 2004
7) Chandrajit L. Bajaj and Valerio Pascucci, *Splitting a Complex of Convex Polytopes In Any Dimension*, ACM Press , 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