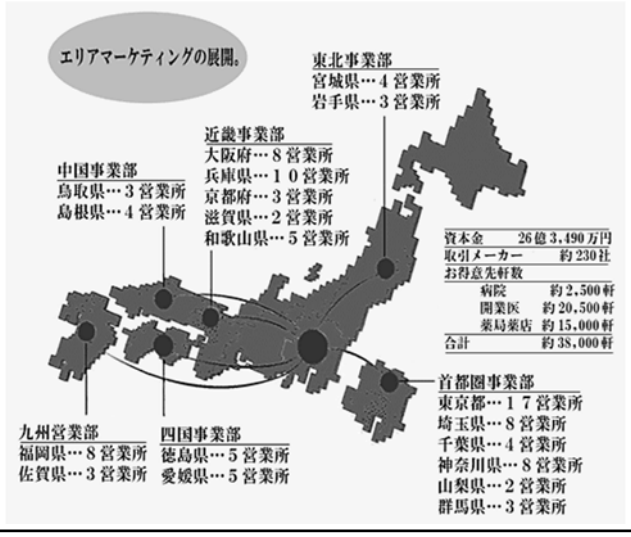
計算幾何学
Computational Geometry

会津大学

第四章　ボロノイ図
Voronoi Diagram

ある会社の営業所分布

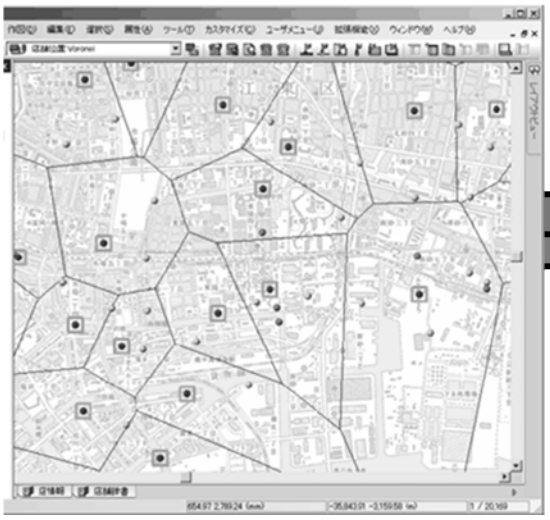


経営戦略の策定

- 仮定
  1. 価格の地域差はない
  2. 単位距離あたりの運搬コストは同じ
- 目標
  - トータルコストを最小化にして、全国範囲内において、顧客にサービスを提供するための最適な分担地域を決定する
- 解答
  - 計算幾何学の最近点問題⇒ボロノイ図

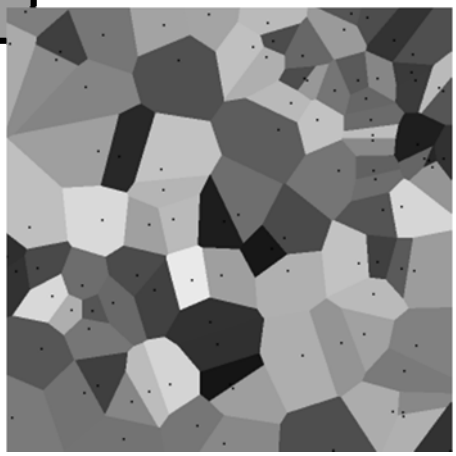
様々な最近点問題

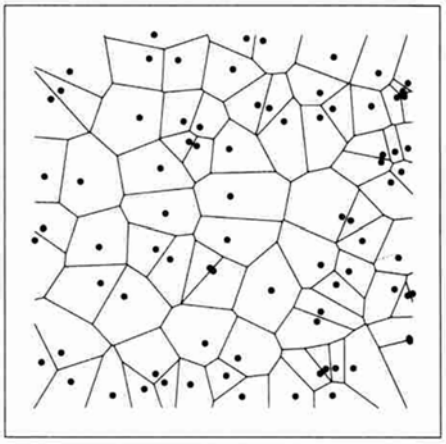
- 通信基地局
- 郵便局
- スーパー
- コンビニ
- レストラン
- 電車駅
- 高速IC
- バス停



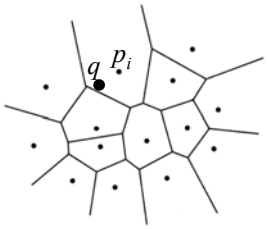
解答→ボロノイ図

- 第一印象⇒蜂の巣、細胞、勢力圏？





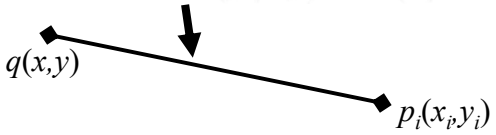
問題の一般化



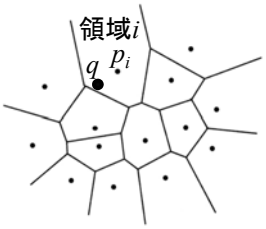
- 平面上のn個の点集

  $P=\{\ p_i(x_i,y_i)\mid i=1, 2, \ldots, n\ \}$とする
- 平面上任意の点$q(x,y)$に対して、qとp_iの2点間距離が最も近い点p_iを求める
- 2点間のユークリッド距離(Euclidean distance)

$$dist(p,q) = \sqrt{(x_i - x)^2 + (y_i - y)^2}$$



定義



- 平面上にn個の異なる点の集合P

$$P = \{\, p_i(x_i, y_i) \mid i = 1, 2, \ldots, n \,\}$$

- 点p_i⇔領域i⇒n個の平面領域分割
- 領域iに含まれる任意点qについて右の関係式が成立

$$dist(q, p_i) < dist(q, p_j),\ j \neq i$$

- 上記性質を持つ平面の領域分割⇒Pのボロノイ図

重要な関係式！
距離の定義は実際問題に依存

様々な距離の定義

http://reference.wolfram.com/language/guide/DistanceAndSimilarityMeasures.html

会津大学

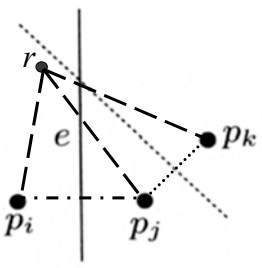
任意次元空間の二点**u**と**v**

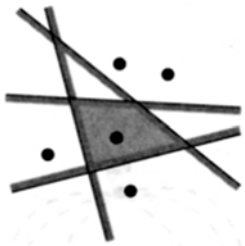
D = diagonal matrix of variance
V = covariance matrix

No	Metric	Definition
1	euclidean	$\sqrt{\sum(\mathbf{u}-\mathbf{v})^2}$
2	seuclidean	$\sqrt{(\mathbf{u}-\mathbf{v})\mathbf{D}^{-1}(\mathbf{u}-\mathbf{v})^T}$
3	mahalanobis	$\sqrt{(\mathbf{u}-\mathbf{v})\mathbf{V}^{-1}(\mathbf{u}-\mathbf{v})^T}$
4	cityblock (manhattan)	$\sum\lvert\mathbf{u}-\mathbf{v}\rvert$
5	minkowski	$\left\{\sum\lvert\mathbf{u}-\mathbf{v}\rvert^P\right\}^{\frac{1}{P}}$
6	cosine	$1-\frac{\mathbf{u}\bullet\mathbf{v}}{\lVert\mathbf{u}\rVert\cdot\lVert\mathbf{v}\rVert}$
7	correlation	$1-\frac{(\mathbf{u}-\bar{\mathbf{u}})\bullet(\mathbf{v}-\bar{\mathbf{v}})}{\lVert\mathbf{u}-\bar{\mathbf{u}}\rVert\cdot\lVert\mathbf{v}-\bar{\mathbf{v}}\rVert}$
8	chebychev (chessboard)	$\max\left(\lvert\mathbf{u}-\mathbf{v}\rvert\right)$
9	canberra	$\sum\lvert\mathbf{u}-\mathbf{v}\rvert/\left(\lVert\mathbf{u}\rVert+\lVert\mathbf{v}\rVert\right)$
10	braycurtis	$\sum\lvert\mathbf{u}-\mathbf{v}\rvert/\sum\lvert\mathbf{u}+\mathbf{v}\rvert$

一つの領域の構造

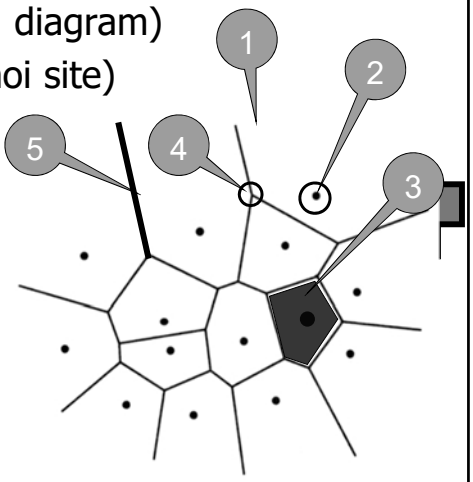
- 2点間の線分の垂直2等分線
- 平面を2つ半平面に分割
  - $h(p_i, p_j) \rightarrow p_i$を含む開半平面
  - $h(p_j, p_i) \rightarrow p_j$を含む開半平面
- 開半平面（Open half plane）
  - 分割線を含まない
- $r \in h(p_i, p_j)$



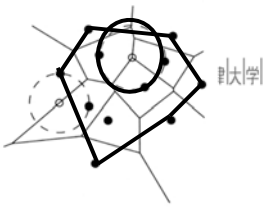


関連術語

1. ボロノイ図(Voronoi diagram)
2. 母点、サイト(Voronoi site)
3. ボロノイ領域 (Voronoi region)
4. ボロノイ頂点 (Voronoi vertex)
5. ボロノイ辺 (Voronoi edge)



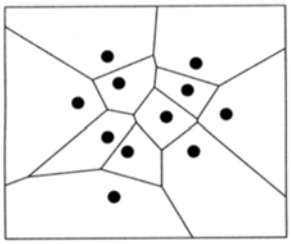
性質



1. ボロノイ領域→凸領域
2. 有界でない（非有界）ボロノイ領域⇔母点から構成される凸包上の点の領域（必要十分条件）
3. ボロノイ頂点→三つのボロノイ辺の共通点
4. ボロノイ領域V(p_1),V(p_2),V(p_3)の頂点を円心として、3母点p_1, p_2, p_3を通る円は他の母点を含まない（ボロノイ頂点は最も近い三つの母点から等距離）
5. 母点p_iが母点p_jの最近隣点であれば、ボロノイ領域V(p_i)とV(p_j)は隣接している、言い換えれば、ボロノイ辺を共有する
6. n点のボロノイ図は高々2n-5個のボロノイ頂点と高々3n-6本のボロノイ辺を持つ
7. ドローネ三角形分割（第5章）と双対関係を持つ

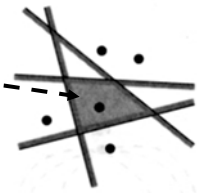
構成方法

1. 直接法→ $O(n^3)$
   - 垂直2等分線の定義による直観的な方法
2. 逐次添加法→ $O(n^2)$
   - 3点から出発、1点ずつ添加していく
3. 分割統治法→ $O(n\log n)$
   - 分割→構成→統合
4. Fortune走査法→ $O(n\log n)$
   - 巧妙な平面走査法

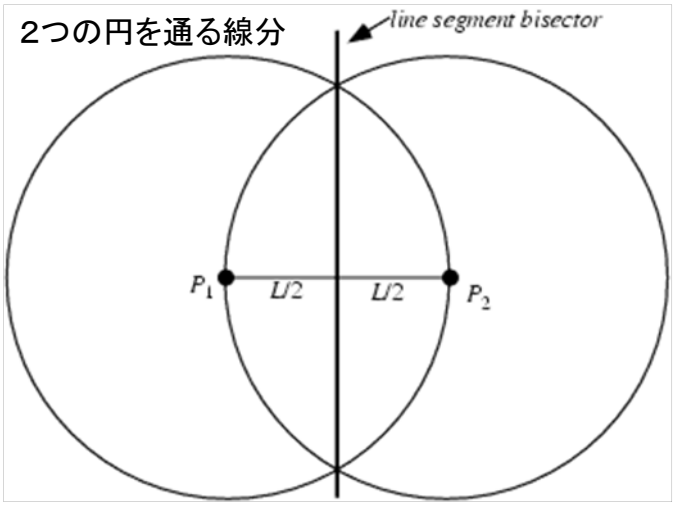


直接法

- 母点p_iとp_jの垂直2等分線は平面を2つ半平面$h(p_i, p_j)$と$h(p_j, p_i)$に分割する
- p_iを含む開半平面$h(p_i, p_j)$における任意の点からは母点p_jよりp_iの方が近いので、定義により、ボロノイ領域は$h(p_i, p_j)$にある
- n-1個の半平面の共通部分（交わり）を求めれば、ボロノイ領域（**灰色の部分**）が得られる

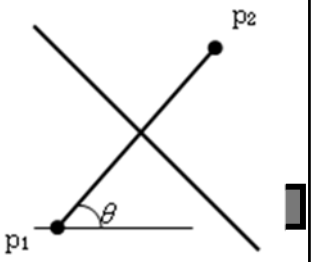


2等分線の求め方ー幾何的方法



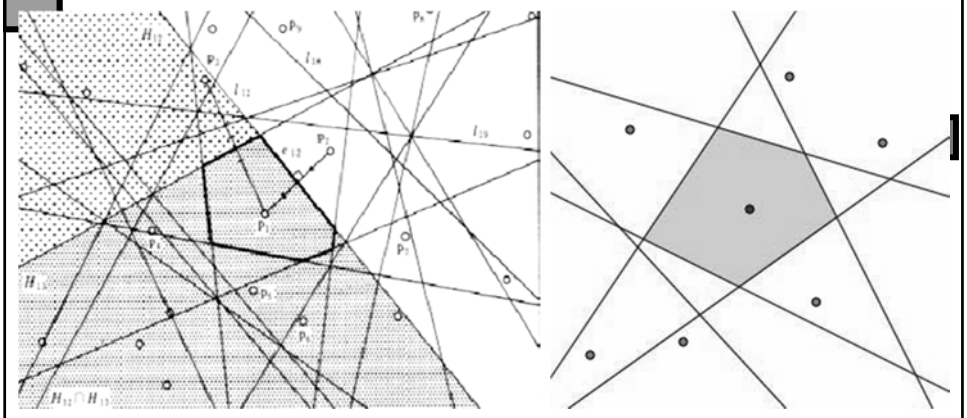
2等分線の求め方ー解析的方法

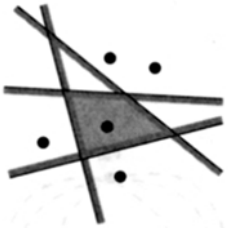
- p_1p_2線分　$y=\alpha x+\beta$
  - $\alpha=(y_2-y_1)/(x_2-x_1)$
  - $\beta=y_1-\alpha x_1=(x_2y_1-x_1y_2)/(x_2-x_1)$
  - $\theta=\mathrm{tg}^{-1}\alpha$
- 垂直2等分線　$y=\alpha' x+\beta'$
  - $\alpha'=\mathrm{tg}(\theta+90^\circ)=-\mathrm{ctg}\theta=-1/\mathrm{tg}\theta=-1/\alpha$
- 交差点で、$y'=y$、$x=(x_1+x_2)/2$
  - $\alpha' x+\beta'=\alpha x+\beta$
  - $\beta'=(\alpha-\alpha')x+\beta=(\alpha-\alpha')(x_1+x_2)/2+\beta$



ーつのボロノイ領域

- たくさんの半平面の共通部分から求めたボロノイ領域

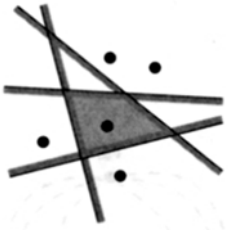




アルゴリズム

- 入力：n点の集合S_n
- 出力：任意点p_kのボロノイ領域$V(p_k)$

1. 初期化
2. for i=1 to n
3. 　if $i \neq k$ then
4. 　　線分 $p_k p_i$ の垂直2等分線を引く
5. 　　p_k側の開半平面$h(p_k, p_i)$を求める
6. 　　$V(p_k) = V(p_k) \cap h(p_k, p_i)$

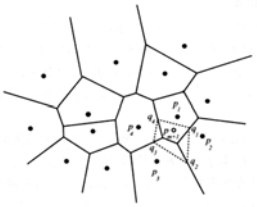


計算量

- 2本の2等分線の交差点を求める
  - 1つ計算単位とする
  - $h(p_1,p_2) \cap h(p_1,p_3)$=1つ計算単位
  - $h(p_1,p_2) \cap h(p_1,p_3) \cap h(p_1,p_4)$=2つ計算単位
  - …
  - $h(p_1,p_2) \cap h(p_1,p_3) \cap h(p_1,p_4) \cap \ldots \cap h(p_1,p_n)$=$n$-2の計算単位
- 1個領域の全部計算量
  - $1+2+\cdots+(n-2)=(n-1)(n-2)/2=O(n^2)$
- n個領域の全部計算量=$O(n^3)$

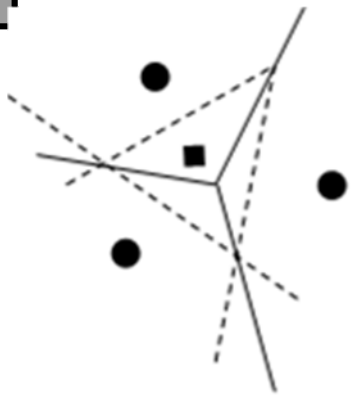
逐次添加法

- まずは3つ母点p_1, p_2, p_3 からスタートして、ボロノイ図を構成する
- 残りの母点を一点ずつ添加していきながら、新たなボロノイ領域を求める
- この新たに作られたボロノイ領域の内部にある元のボロノイ辺の一部を消せば、新たなボロノイ図になる
- 全ての母点について処理する

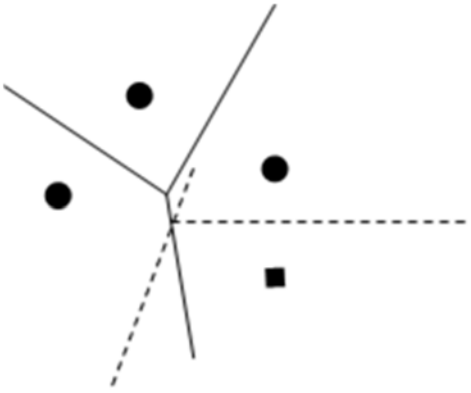


新領域＝2つ可能性

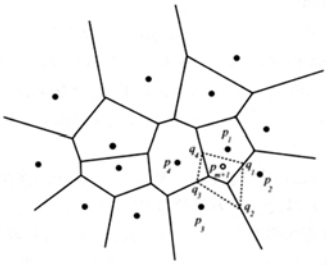
有界である例



有界でない（非有界）例

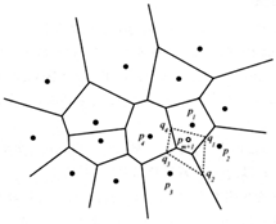


アルゴリズム



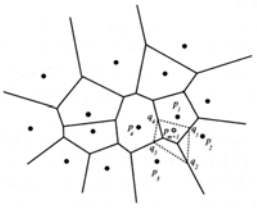
1. 任意3点のボロノイ図を構成
2. for (m=3 to n-1) {
3. 既存母点$p_1, p_2, \ldots, p_m$から新たな添加母点p_{m+1}と一番近い母点$p_{(1)}$を求める　（点位置決定問題）
4. 線分　　の垂直2等分線とボロノイ領域$V_m(p_{(1)})$の辺との交点q_1を求める
5. 交点q_1のある辺と隣接している$V_m(p_{(1)})$以外のボロノイ領域を$V_m(p_{(2)})$とする
6. k=2

アルゴリズム続き



7. while ($p_{(k)} \neq p_{(1)}$) do {
8. 線分　　の垂直2等分線とボロノイ領域$V_m(p_{(k)})$の辺との新たな交点q_kを求める
9. 交点q_kのある辺と隣接している$V_m(p_{(k)})$以外のボロノイ領域を$V_m(p_{(k+1)})$とする
10. $k = k + 1$
11. } // end of while
12. 交点の集合q_i（i = 1, 2,…, k -1）を順番に繋いで多角形$q_1 q_{2\ldots} q_{k-1}$を構成する。この多角形は母点p_{m+1}のボロノイ領域$V_{m+1}(p_{m+1})$になる
10. } // end of for

計算量

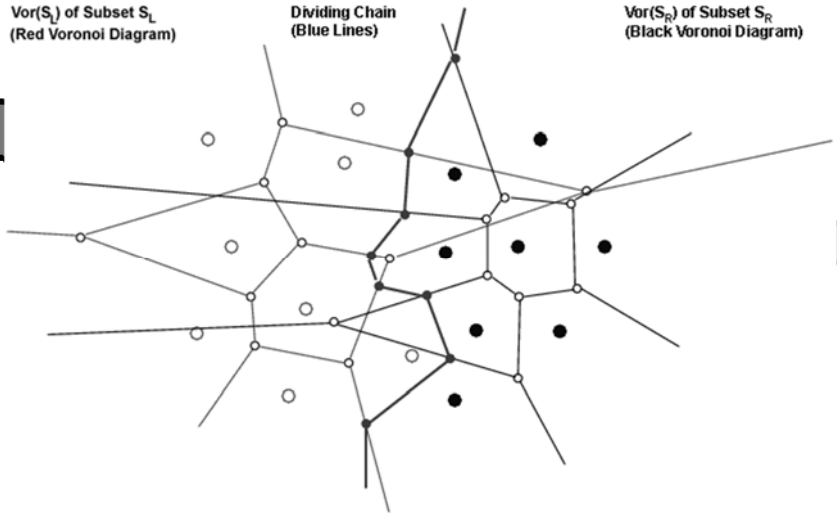


- 2点間の2等分線を求める
  - 1つ計算単位とする
  - 4点目→3つ計算単位
  - 5点目→4つ計算単位
  - …
  - n点目→n-1の計算単位
- n点全部の計算量
  - $3+4+\cdots+(n-1)=(n+2)(n-3)/2=O(n^2)$

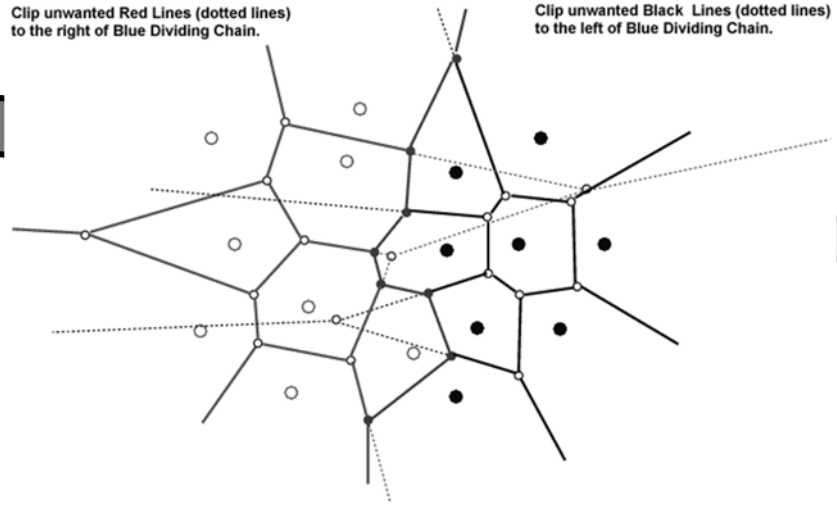
分割統治法

- **Divide-and-Conquer** is one of the fundamental paradigms for designing efficient algorithms
- The original problem is recursively divided into several simpler sub-problems of roughly equal size, and the solution of the original problem obtained by merging the solutions of the sub-problems.
- When being applied in Voronoi diagram, the set of sites, S, is split up by a **dividing line** (分割線) into subsets S_L and S_R of about same sizes. Then, Voronoi diagram, $\mathrm{Vor}(S_L)$, of subset S_L and Voronoi diagram, $\mathrm{Vor}(S_R)$, of subset S_R are computed recursively.
- Shamos and Hoey, the first $O(n\log n)$ algorithm, 1975

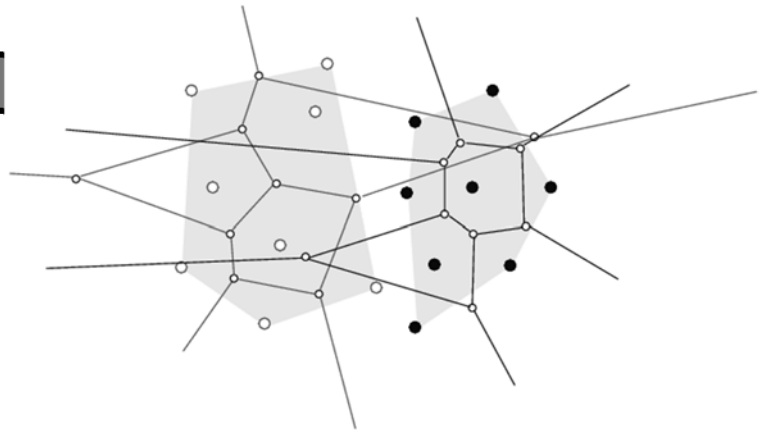
左右ボロノイ図から分割線構成



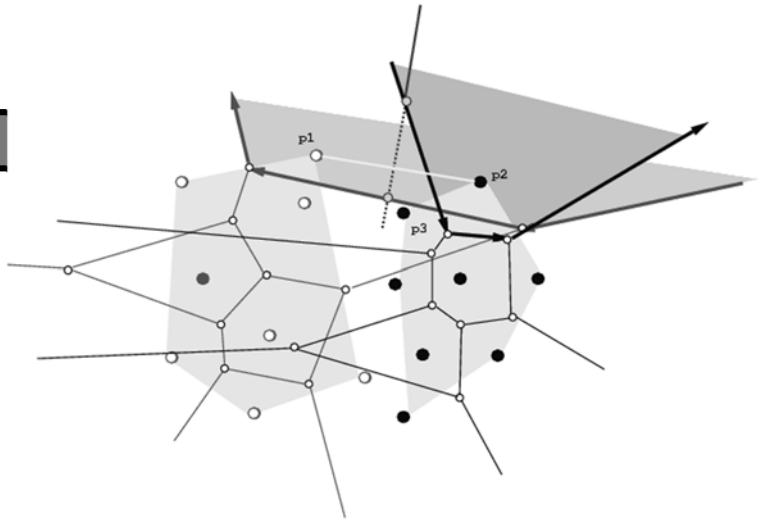
統合⇒不要線分の削除



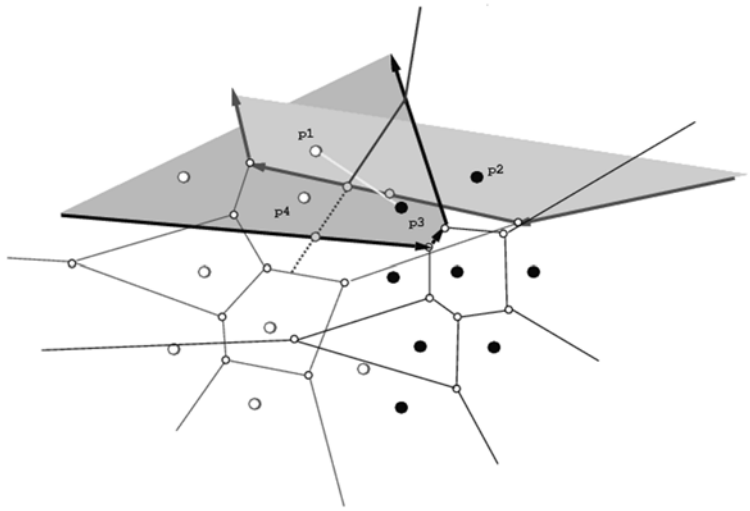
分割線の構成—Step 1



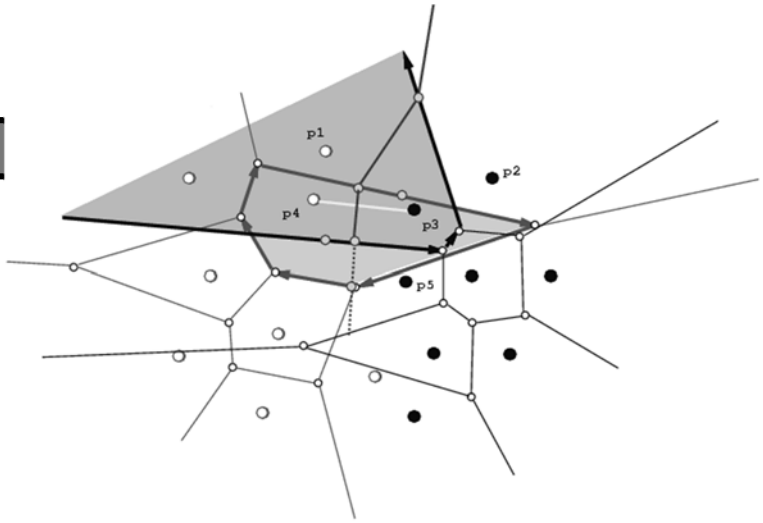
分割線の構成—Step 2

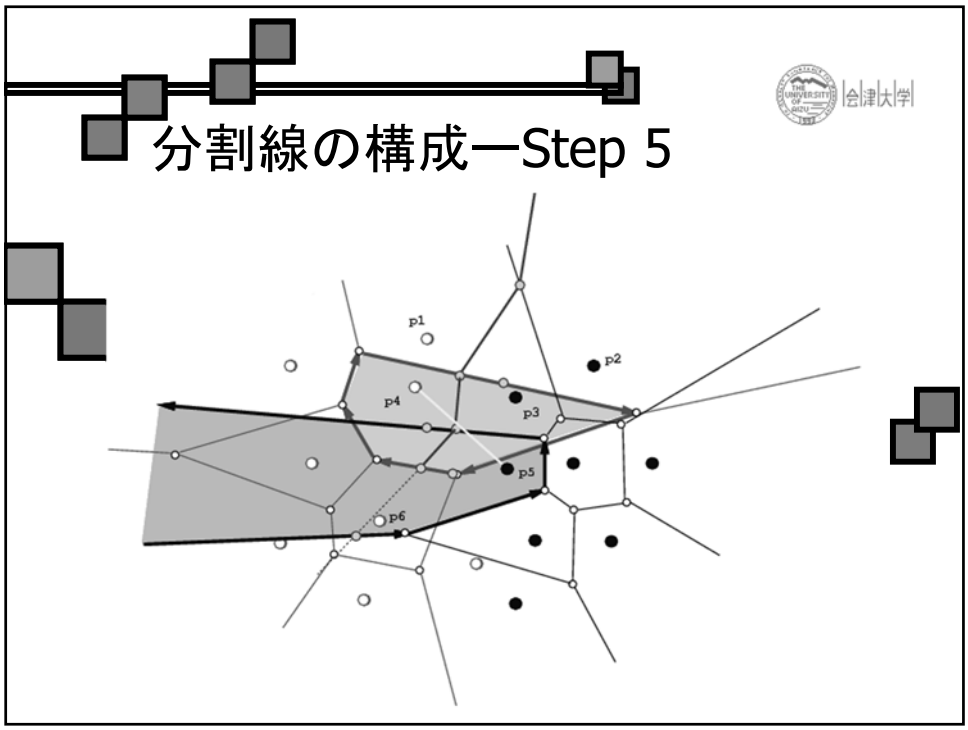


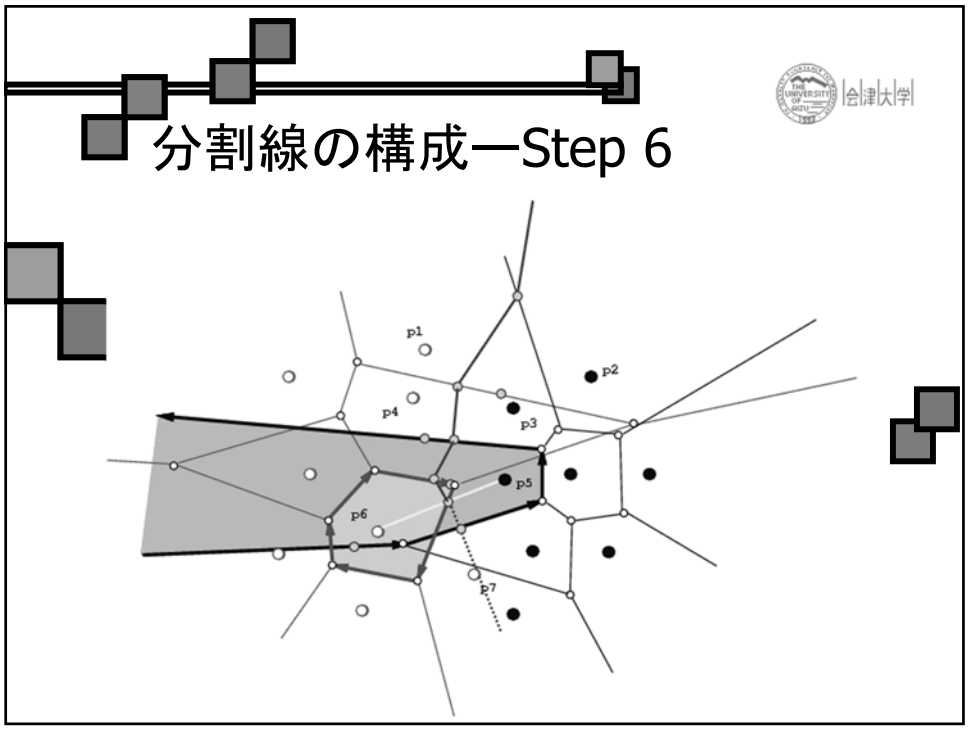
分割線の構成—Step 3

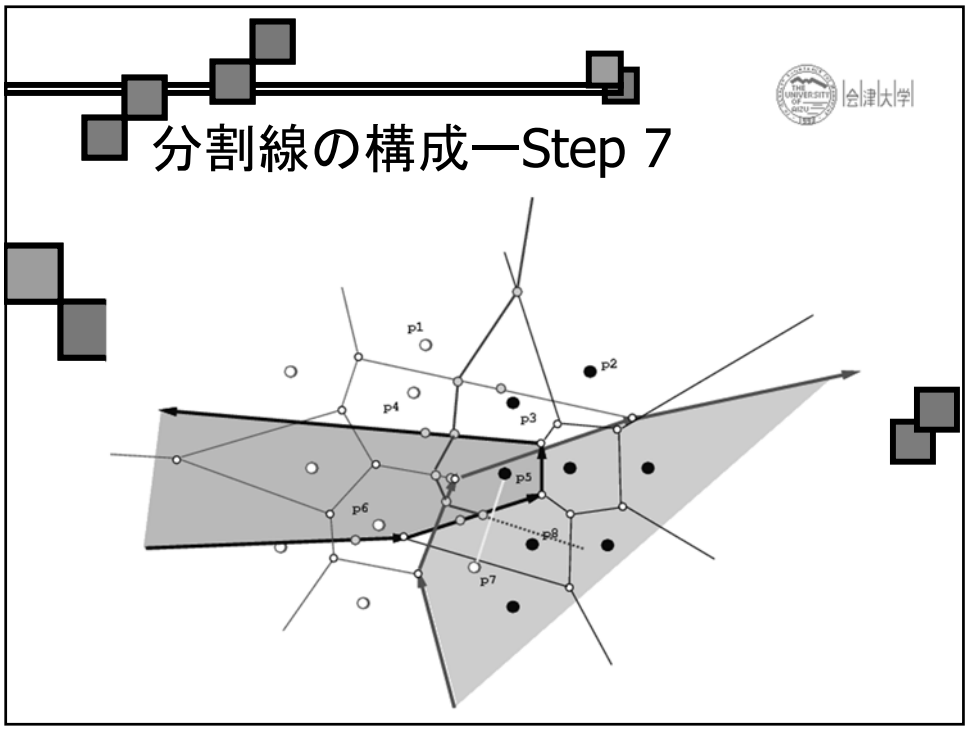


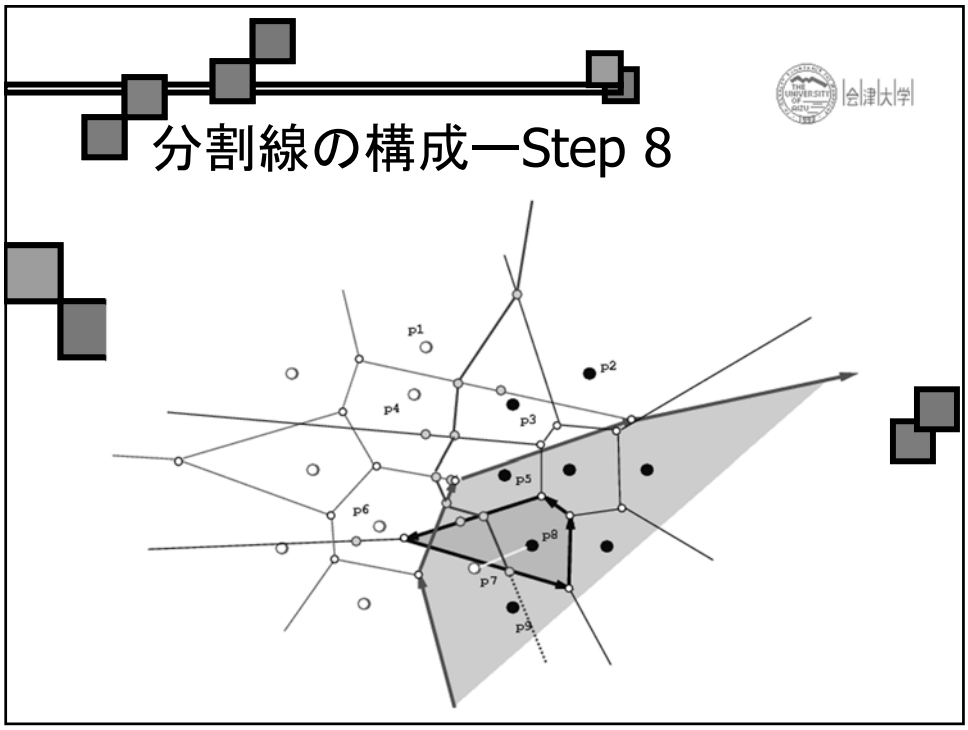
分割線の構成—Step 4

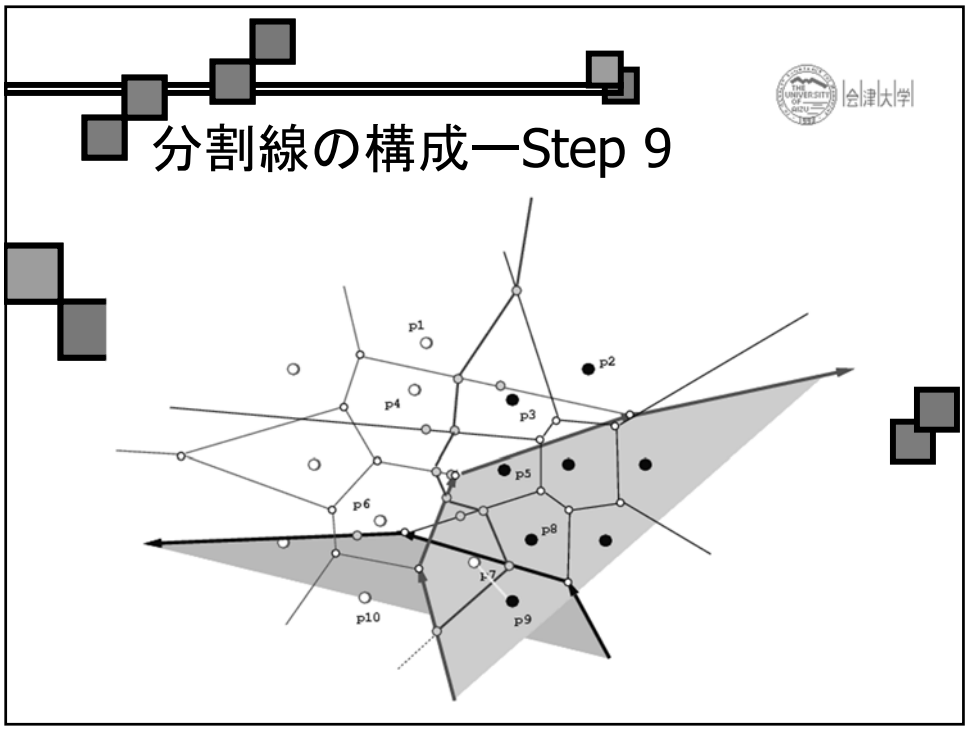


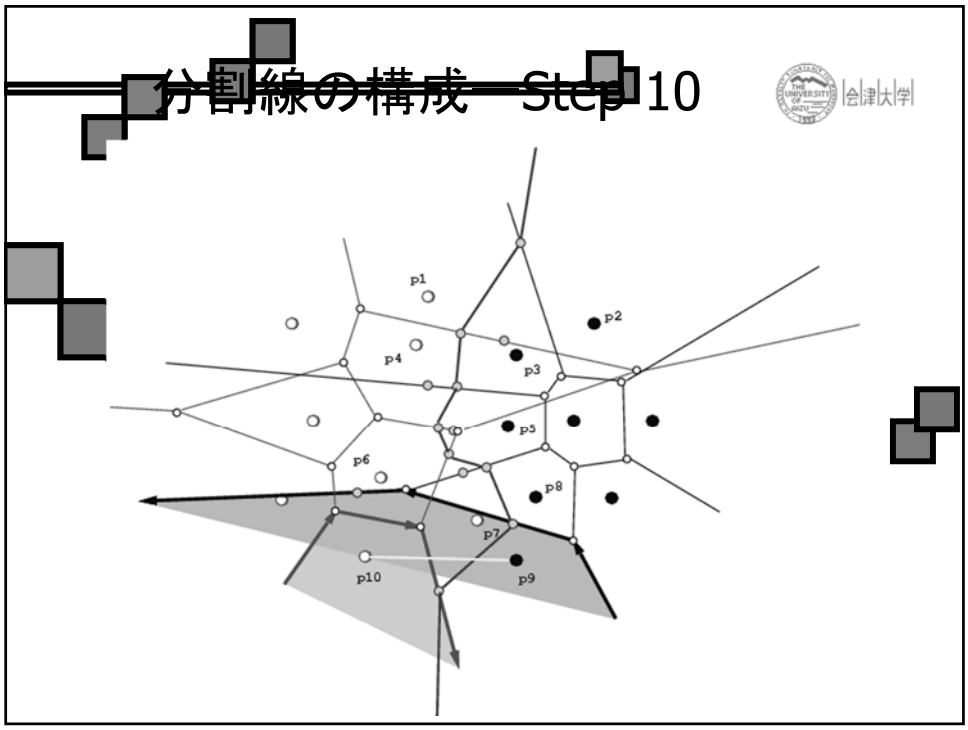

分割線の構成ーStep 5
p1
p2
p3
p4
p5
p6


分割線の構成ーStep 6
p1
p2
p3
p4
p5
p6
p7

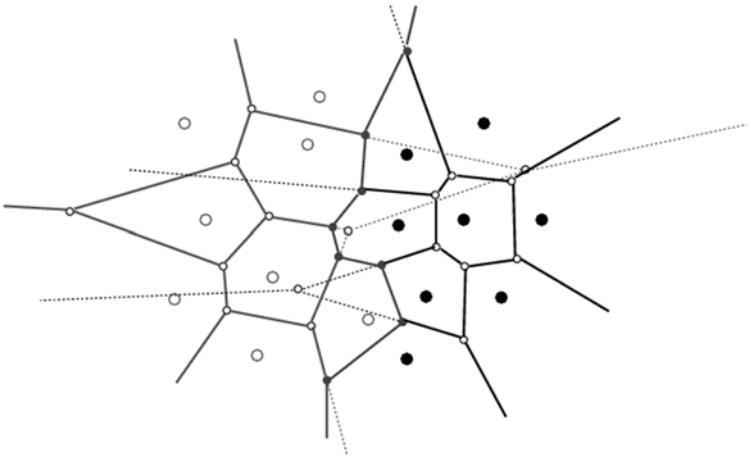

分割線の構成ーStep 7
p1
p2
p3
p4
p5
p6
p7
p8


分割線の構成ーStep 8
p1
p2
p3
p4
p5
p6
p7
p8
p9

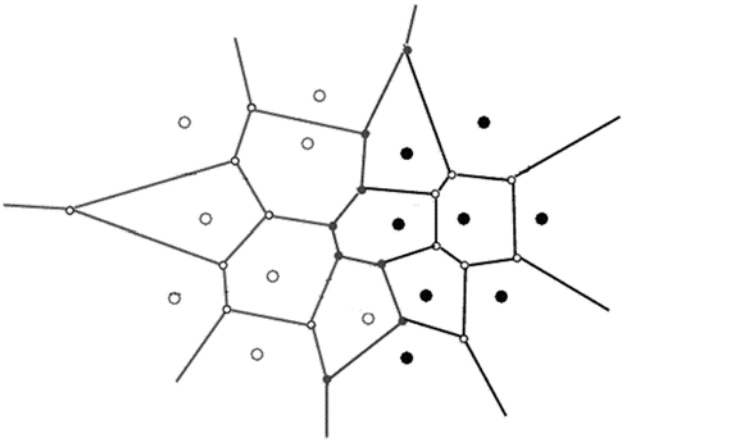

分割線の構成—Step 9
p1
p2
p3
p4
p5
p6
p7
p8
p9
p10


分割線の構成—Step 10
p1
p2
p3
p4
p5
p6
p7
p8
p9
p10

左右ボロノイ図の統合ーStep 1



左右ボロノイ図の統合ーStep 2



アルゴリズム

メインルーチン

- VORONOI_DIAGRAM
- 入力：平面上の母点集合$P=\{p_1,p_2,\ldots,p_n\}$, x座標の昇順
- 出力：ボロノイ図$Vor(P)$

1. t=integer(n/2), let $P_L=\{p_1,p_2,\ldots,p_t\}$, $P_R=\{p_{t+1},p_{t+2},\ldots,p_n\}$
2. 回帰的にP_Lのボロノイ図$Vor(P_L)$を構成する
3. 回帰的にP_Rのボロノイ図$Vor(P_R)$を構成する
4. MERGE_VORONOIを用いて$Vor(P_L)$と$Vor(P_R)$を統合し、$Vor(P)$を構成する
5. $Vor(P)$を返す

アルゴリズム(1/3)

コアルーチン

- MERGE_VORONOI
- 入力：ボロノイ図$Vor(P_L)$と$Vor(P_R)$
- 出力：ボロノイ図$Vor(P)$、$P=P_L \cup P_R$

1. P_LとP_Rの凸包を構成する
2. UPPER_COMMON_SUPPORTを用いてupper common support line $U(P_L, P_R)$ を探索する
3. 母点p_L(P_Lに属する)と母点p_R(P_Rに属する)の垂直二等分線$B(P_L, P_R)$上にある上向き方向の無限遠点$\rightarrow w_0$
4. $0 \rightarrow i$

アルゴリズム(2/3)

5. While $U(P_L, P_R)$!= lower common support line
   - ① $i \leftarrow i+1$
   - ② $B(P_L, P_R)$と$Vor(P_L)$の境界線との交差点a_Lを探索する
   - ③ $B(P_L, P_R)$と$Vor(P_R)$の境界線との交差点a_Rを探索する
   - ④ If $a_L.y > a_R.y$ then
     - $w_i \leftarrow a_L$
     - $p_L \leftarrow a_L$の所在するボロノイ辺の反対側にある母点
     - else
     - $w_i \leftarrow a_R$
     - $p_R \leftarrow a_R$の所在するボロノイ辺の反対側にある母点

アルゴリズム(3/3)

6. $m \leftarrow i$
7. w_{m+1}←母点p_L(P_Lに属する)と母点p_R (P_Rに属する)の垂直二等分線$B(P_L, P_R)$上にある下向き方向の無限遠点
8. polygonal line (w_0w_1, w_1w_2,..., w_mw_{m+1})を加える
9. $Vor(P_L)$からpolygonal lineの右部分を削除する
10. $Vor(P_R)$からpolygonal lineの左部分を削除する
11. $Vor(P_L)$と$Vor(P_R)$を統合したボロノイ図$Vor(P)$を返す

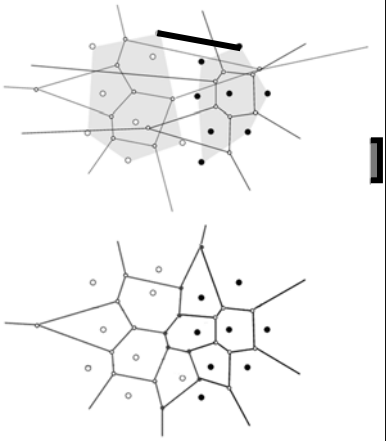
アルゴリズム

- UPPER_COMMON_SUPPORT
- 入力：2つ凸多角形CP_LとCP_R、CP_Lは完全にCP_Rの左側
- 出力：頂点対u(CP_Lに属する)とv(CP_Rに属する), $U(u, v)$ =凸多角形CP_LとCP_Rのupper common support line

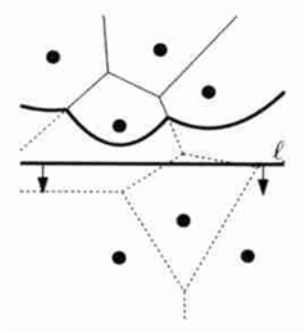
1. 最大x座標を持つ頂点u(CP_Lに属する)と最小x座標を持つ頂点v(CP_Rに属する)を探索する
2. While next[u] > $U(u, v)$, u←next[u]
3. While next[v] > $U(u, v)$, v←next[v]
4. Return $U(u, v)$

計算量

- UPPER_COMMON_SUPPORT
  - $O(n)$
- MERGE_VORONOI
  - 左右ボロノイ図にある関連母点の垂直2等分線の計算
  - $O(n)$
- VORONOI_DIAGRAM
  - 分割線の計算と左右ボロノイ図の統合
  - $T(n)=2T(n/2)+O(n)$ $=O(n\log n)$



平面走査法?



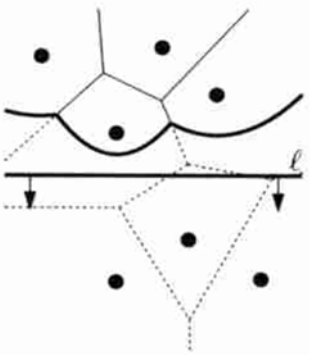
- 既知：イベント点←母点
- 求め：構造情報→ボロノイ辺
- 線分交差と同様な平面走査法が使用可能?
- 水平な走査線 *l* を平面上に上から下へ動かしていきながら、ボロノイ辺を求められる?
- 結論：大変！
  ⇒ボロノイ辺は走査線 *l* の上の母点（既知）だけでなく、下の母点（未知）にも依存する！

Fortune走査法

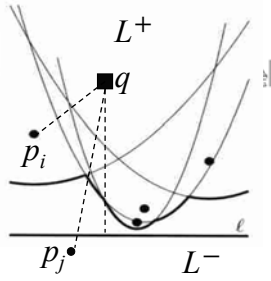
走査線 *l* とボロノイ辺の交差状況を管理するのではなく、走査線 *l* の下側にある母点によって***変えられない***ように走査線 *l* の上側にある母点だけを持って決められる***部分的***なボロノイ図の構造情報を管理する

変えられないもの？⇒ボロノイ領域の性質→母点との距離＝最短

発想の転換⇒完璧より折衷＝妥協の哲学



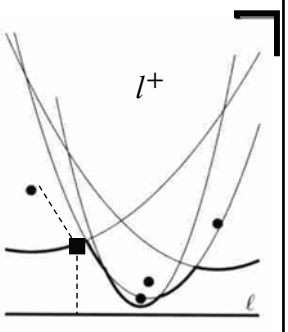
基本的な考え方



- L^+←lの上側の閉半平面
- L^-←lの下側の開半平面
- L^-の母点によって***変えられない***L^+のボロノイ領域の境界範囲？⇔ L^+の他の母点より、ある母点p_iに最も近い点qの***最大範囲***⇒ボロノイ領域$V(p_i)$の***一部***
- L^+の点q→ dist($q \in L^+$,$p_j \in L^-$) > dist($q \in L^+$,l)
- L^+の点q→ dist($q \in L^+$,$p_i \in L^+$)<=>dist($q \in L^+$,l)
  1. "="の場合→放物線の弧によって定められる境界
  2. "<"の場合→放物線の弧の上側
  3. ">"の場合→放物線の弧の下側

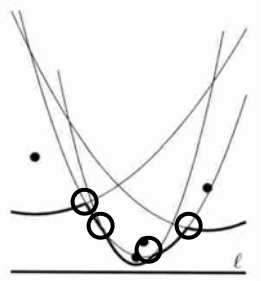
ビーチライン(beach line)

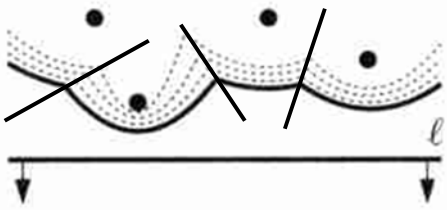
- 走査線 l までの距離とL^+平面の各母点との距離が等しい点は、それぞれの放物線になる
- これらの放物線の弧の系列→beach line
- 母点p_i (p_{ix}, p_{iy})の放物線方程式
  l_y→走査線のy座標



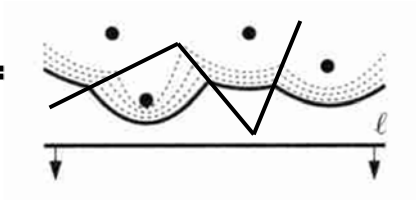
ブレークポイント(breakpoint)

- ビーチラインを構成している各母点の放物線の弧の交点→ breakpoint
- 走査線が上から下へ動く間に、ブレークポイントはボロノイ辺を辿っている！⇒不思議？





考察



- 走査線を動かしながら、ビーチラインを管理すれば、ボロノイ辺が分る
- ビーチラインの構造変化：Where? How?

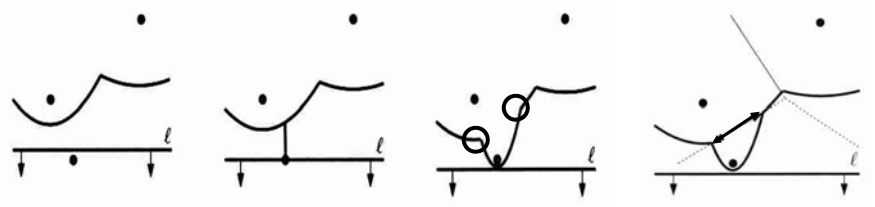
  ①新たな放物線の弧が現れる

  ②放物線の弧が1点に縮小し消えていく
- 走査線の連続移動の代わり、イベント点のところだけを考えてみる
  - ビーチラインの構造変化とイベント点の関係？

サイトイベント(site event)

- 新たな放物線の弧がビーチラインに現れる

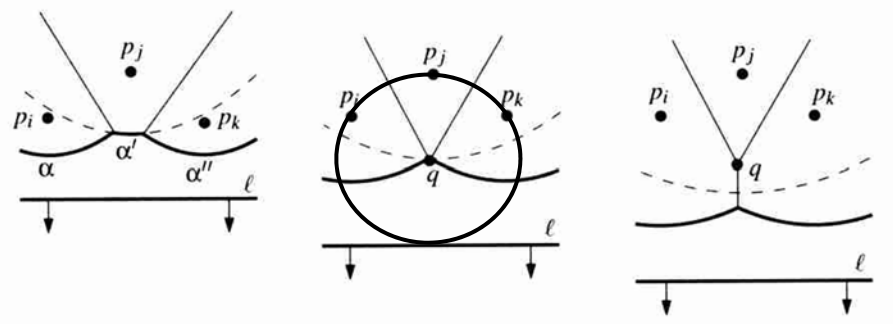
1. 走査線が新たな母点に到達した時（左図中央）
2. 新たなブレークポイントが2つ現れる（左図右）
3. 最初、2つブレークポイントは同一点に統合されるが、走査線の移動と伴って段々異なる方向に向かって新たなボロノイ辺をたどっていく（右図）



円イベント(circle event)

- 放物線の弧が1点に縮小し消えていく

1. 点qから3母点p_i, p_j, p_k及び走査線lまで等距離
2. 点qはボロノイ頂点←$(p_x\text{-}q_x)^2+(p_y\text{-}q_y)^2=R^2$



まとめ

- ビーチラインの構造変化：Where? How?

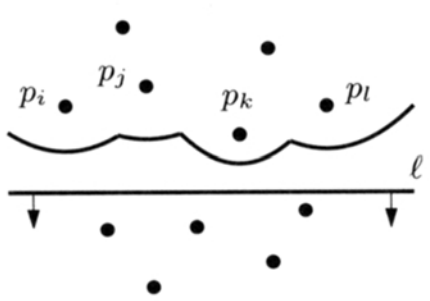
1. サイトイベント
   - ① 新たな弧が現れる
   - ② 新たなボロノイ辺が成長する
2. 円イベント
   - ① 本来存在した弧が消える
   - ② 成長している2つ辺が交差してボロノイ頂点を形成する

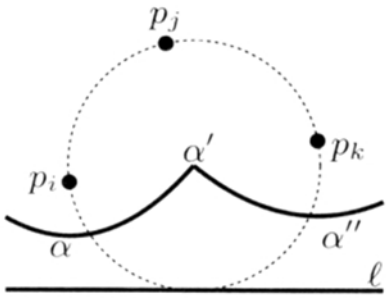
データ構造

- Fortune走査法の基本原理を解明したら、実装するには？
- 走査線を上から下へ動かしている間に必要な情報を管理するための適切なデータ構造が必要！→3種類のデータ構造
  1. イベント
  2. 走査線状態
  3. ボロノイ図

イベントのデータ構造

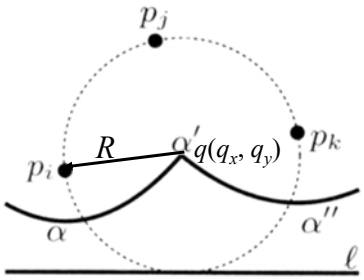
- サイトイベント→事前に分っている
  →y座標順でリストに保存
- 円イベント　→事前に分らない→検出方法？





円イベントの検出方法

- ビーチラインから弧が消えることによって生じた
- 三つの母点p_i, p_j, p_kによって定められる円はその最下点が走査線上にある時→ p_jに対応する弧α’が一点に集中し消えていく→ボロノイ頂点



$(p_x-q_x)^2+(p_y-q_y)^2=R^2$

2次曲線と円の方程式

- 一般的な2次曲線　$ax^2+bxy+cy^2+dx+ey+f=0$
- 円の場合　$a=c$ and $b=0$
- 円の方程式　$a(x^2+y^2)+dx+ey+f=0$

$$\begin{pmatrix} x^2+y^2 & x & y & 1 \end{pmatrix}\begin{pmatrix} a \\ d \\ e \\ f \end{pmatrix}=0$$

$$\left(x+\frac{d}{2a}\right)^2+\left(y+\frac{e}{2a}\right)^2+\left(\frac{f}{a}-\frac{d^2+e^2}{4a^2}\right)=0$$

$$x_0=\frac{-d}{2a} \qquad y_0=\frac{-e}{2a} \qquad R=\sqrt{\frac{d^2+e^2}{4a^2}-\frac{f}{a}}$$

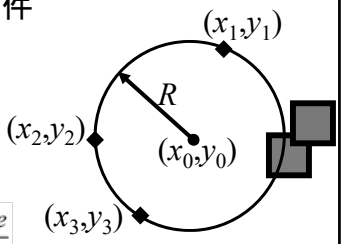
$$(x-x_0)^2+(y-y_0)^2=R^2$$

円心と半径の計算方法

- 円の方程式　$(x-x_0)^2+(y-y_0)^2=R^2$
  - 既知3点を代入して解を求める
  - a, d, e, f は非零解の必要十分条件

$$\begin{vmatrix} x^2+y^2 & x & y & 1 \\ x_1^2+y_1^2 & x_1 & y_1 & 1 \\ x_2^2+y_2^2 & x_2 & y_2 & 1 \\ x_3^2+y_3^2 & x_3 & y_3 & 1 \end{vmatrix}=0$$

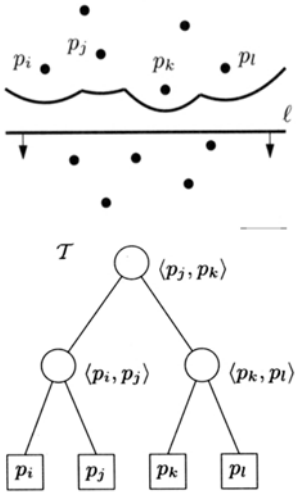
$$R=\sqrt{\frac{d^2+e^2}{4a^2}-\frac{f}{a}} \qquad x_0=\frac{-d}{2a} \qquad y_0=\frac{-e}{2a}$$



$$a=\begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix} \quad d=-\begin{vmatrix} x_1^2+y_1^2 & y_1 & 1 \\ x_2^2+y_2^2 & y_2 & 1 \\ x_3^2+y_3^2 & y_3 & 1 \end{vmatrix} \quad e=\begin{vmatrix} x_1^2+y_1^2 & x_1 & 1 \\ x_2^2+y_2^2 & x_2 & 1 \\ x_3^2+y_3^2 & x_3 & 1 \end{vmatrix} \quad f=-\begin{vmatrix} x_1^2+y_1^2 & x_1 & y_1 \\ x_2^2+y_2^2 & x_2 & y_2 \\ x_3^2+y_3^2 & x_3 & y_3 \end{vmatrix}$$

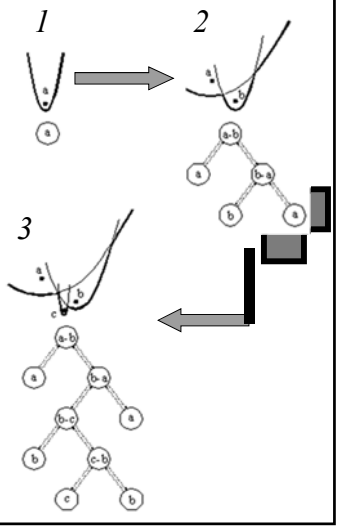
走査線状態のデータ構造

- ビーチライン→放物線の弧→明示に保存しない→母点とブレークポイント→2分探索木$\mathcal{T}$
- $\mathcal{T}$の外点→各弧と対応する母点（x座標順）
- $\mathcal{T}$の内点→各弧のブレークポイント（母点の順序対<p_i, p_j>）
  - p_i＝ブレークポイントの左にある放物線を定義する母点
  - p_j＝ブレークポイントの右にある放物線を定義する母点



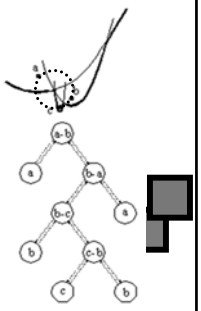
走査線状態木へ新弧の挿入

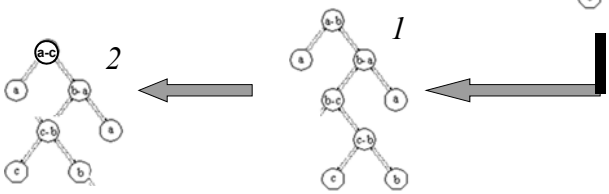
- サイトイベント→新弧→部分木で葉を置き換える
- 右図はビーチラインと走査線状態木の対応関係
  1. サイトイベント a のみの場合
  2. サイトイベント b が現れる時
  3. サイトイベント c が現れる時



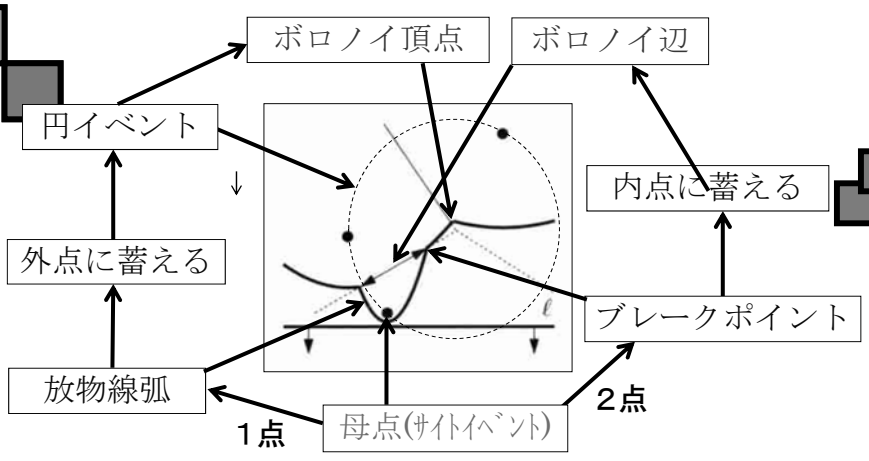
走査線状態木から既存弧の削除

1. 円イベント→既存弧が消える→木から葉*b*を削除する
2. この弧に対応する内点*b*も削除、その左弧*a*と右弧*c*が融合し、弧*a*と弧*c*による新たなブレークポイント<*a*, *c*>を形成する



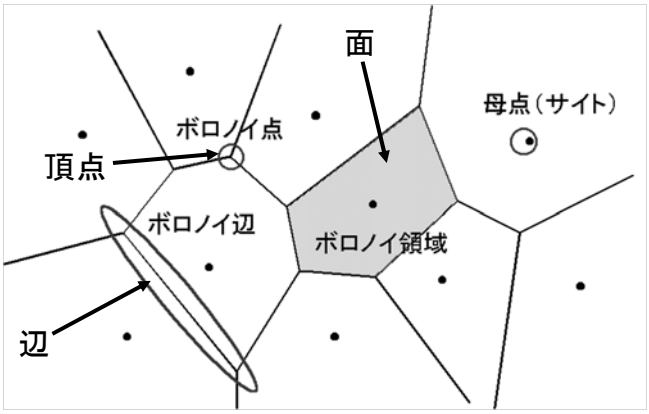


ｲﾍﾞﾝﾄのﾃﾞｰﾀ構造と走査線状態のﾃﾞｰﾀ構造の関連付け



ボロノイ図のデータ構造

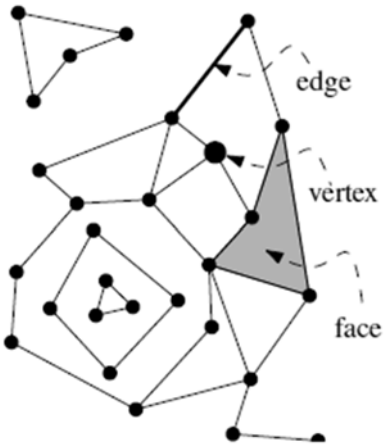
- 全てのボロノイ頂点を含む十分に大きな長方形
  - 平面領域分割⇒3つのﾃﾞｰﾀ構造（辺、頂点、面）



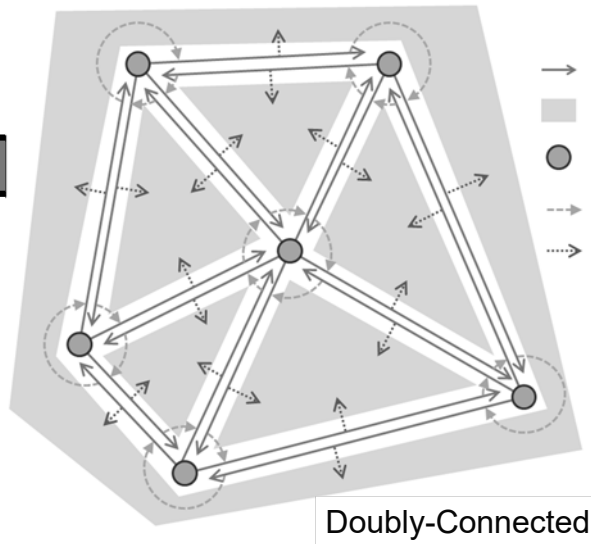
平面領域分割

- 3つのﾃﾞｰﾀ構造
  （辺、頂点、面）
- 二重連結辺リスト
- Doubly-Connected Edge List

a subdivision representation structure with an object for every vertex, every half-edge, and every face



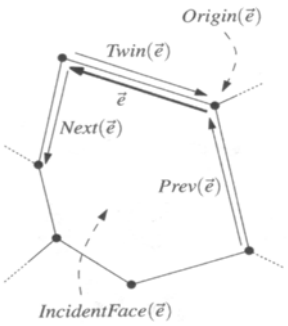
二重連結辺リスト(DCEL)

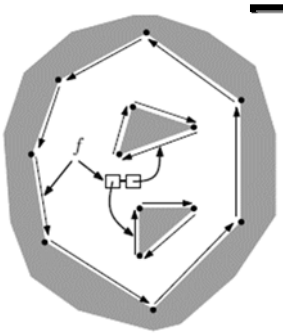


Doubly-Connected Edge List

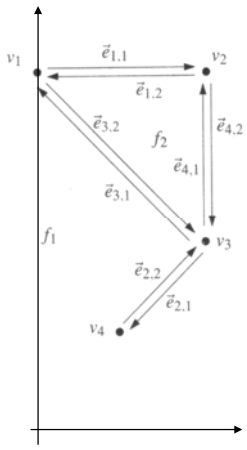
DCEL:3種類の構造体:頂点、面、辺

1. Vertex v ={① 座標、② vを始点とする片辺へのﾎﾟｲﾝﾀ}
2. Face f ={① InnerComponent
   =内側境界上の片辺へのﾎﾟｲﾝﾀ
   ② OuterComponent
   =外側境界上の片辺へのﾎﾟｲﾝﾀ}
3. Edge e ={① 始点へのﾎﾟｲﾝﾀ、
   ② 逆方向へのﾎﾟｲﾝﾀ、
   ③ 接続面へのﾎﾟｲﾝﾀ、
   ④ 次の辺へのﾎﾟｲﾝﾀ、
   ⑤ 前の辺へのﾎﾟｲﾝﾀ}





頂点、面、辺の構造体-実例

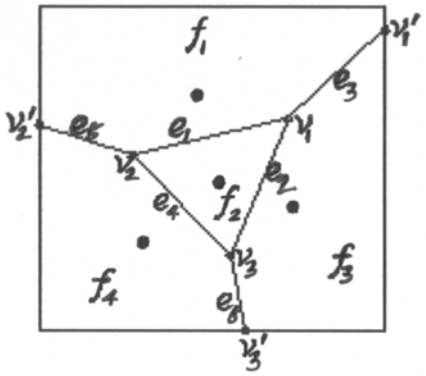


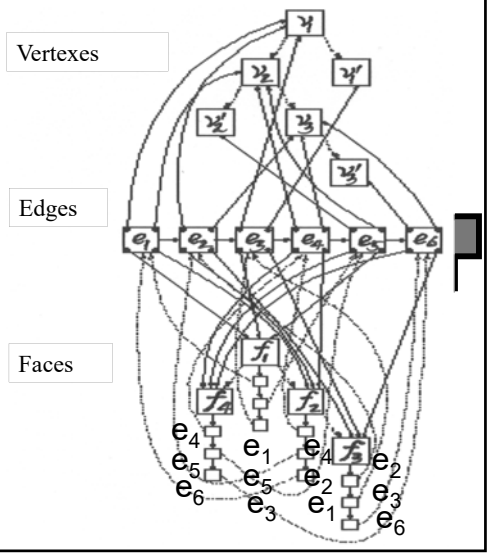
Vertex	Coordinates	IncidentEdge
v_1	$(0,4)$	$\vec{e}_{1,1}$
v_2	$(2,4)$	$\vec{e}_{4,2}$
v_3	$(2,2)$	$\vec{e}_{2,1}$
v_4	$(1,1)$	$\vec{e}_{2,2}$

Face	OuterComponent	InnerComponents
f_1	**nil**	$\vec{e}_{1,1}$
f_2	$\vec{e}_{4,1}$	**nil**

Half-edge	Origin	Twin	IncidentFace	Next	Prev
$\vec{e}_{1,1}$	v_1	$\vec{e}_{1,2}$	f_1	$\vec{e}_{4,2}$	$\vec{e}_{3,1}$
$\vec{e}_{1,2}$	v_2	$\vec{e}_{1,1}$	f_2	$\vec{e}_{3,2}$	$\vec{e}_{4,1}$
$\vec{e}_{2,1}$	v_3	$\vec{e}_{2,2}$	f_1	$\vec{e}_{2,2}$	$\vec{e}_{4,2}$
$\vec{e}_{2,2}$	v_4	$\vec{e}_{2,1}$	f_1	$\vec{e}_{3,1}$	$\vec{e}_{2,1}$
$\vec{e}_{3,1}$	v_3	$\vec{e}_{3,2}$	f_1	$\vec{e}_{1,1}$	$\vec{e}_{2,2}$
$\vec{e}_{3,2}$	v_1	$\vec{e}_{3,1}$	f_2	$\vec{e}_{4,1}$	$\vec{e}_{1,2}$
$\vec{e}_{4,1}$	v_3	$\vec{e}_{4,2}$	f_2	$\vec{e}_{1,2}$	$\vec{e}_{3,2}$
$\vec{e}_{4,2}$	v_2	$\vec{e}_{4,1}$	f_1	$\vec{e}_{2,1}$	$\vec{e}_{1,1}$

頂点、面、辺の構造体間の連携





アルゴリズム

- 入力：平面上の母点集合$P=\{p_1,p_2,\ldots,p_n\}$
- 出力：ボロノイ図$Vor(P)$→DCELで表現した有界長方形の平面領域分割D

1. ｲﾍﾞﾝﾄｷｭｰQ初期化＝すべてのｻｲﾄｲﾍﾞﾝﾄを入れる
2. 走査線状態木TとDCELの平面領域分割Dを空にする
3. while $Q \neq$ nil
4. 　do Qからy_{max}座標を持つイベントを取り出す
5. 　　if このイベントはサイトp_iで生じるサイトイベント
6. 　　then **HandleSiteEvent**(p_i)
7. 　　else **HandleCircleEvent**(γ),

[γ=消えようとする弧と対応するTの葉]

アルゴリズム　続き

8. T内にあるすべての内部節点は、ボロノイ図の片無限辺に対応している
9. 既存すべての頂点を囲む有界長方形を求める
10. 片無限辺を有界長方形に接続するように、DCELを更新する
11. DCELの片辺をたどって、面の構造体、および関係する双方向のﾎﾟｲﾝﾀを追加する

HandleSiteEvent (p_i)

1. If $\mathcal{T}$= *empty* then {p_iを$\mathcal{T}$に挿入、*return*} else $\mathcal{D}o$ *2*〜*5*
2. $\mathcal{T}$からp_iの垂直方向の上にある弧αを探索する。もし、αと対応する葉はQ中の円ｲﾍﾞﾝﾄへのﾎﾟｲﾝﾀを持っていれば、この円ｲﾍﾞﾝﾄは余計なもので、Qから削除
3. 弧αの対応する$\mathcal{T}$の葉を、3個の葉を持つ部分木で置き換える。真中の葉には新たな母点p_iを蓄え、両側2つ葉には、p_jを蓄える（元々のαと対応する）。また、2つ新しい内点に2つ新たなﾌﾞﾚｰｸﾎﾟｲﾝﾄ<p_j, p_i>と<p_i, p_j>を蓄える。必要なら、$\mathcal{T}$ を平衡化する

HandleSiteEvent (p_i)続き

4. 新しい片辺構造体を生成し、p_iとp_jのボロノイ領域を分割する
5. 新たな弧を含んだ連続的となる三つの弧を調べ、2つ処理を行う
   1. 新たな弧を左端とし、右側の2つ弧を含んで、3連続弧を形成する場合→もし2つブレークポイントが最終的に一点に集中する→円イベントをQの中に挿入する、Tの節点とQの節点の間のﾎﾟｲﾝﾀを追加する。
   2. 新たな弧を右端とし、左側の2つ弧を含んで、3連続弧を形成する場合→ステップ5-1と同じ処理を行う

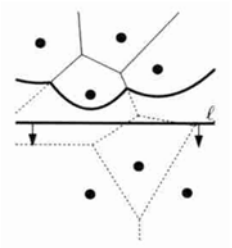
HandleCircleEvent (γ)

1. 消えようとする弧αと対応する葉γを$\mathcal{T}$から削除する。対応する内点（ﾌﾞﾚｰｸﾎﾟｲﾝﾄ）も更新する。必要なら、$\mathcal{T}$を平衡化する。αと関連するすべての円イベントをQから削除する
2. この円ｲﾍﾞﾝﾄの円心を新しい頂点構造体として$DCEL$に追加する。ﾋﾞｰﾁﾗｲﾝの新しいﾌﾞﾚｰｸﾎﾟｲﾝﾀと対応する2つ片辺構造体を生成する。その間のﾎﾟｲﾝﾀも追加する。この新しい頂点を終点とする片辺構造体に3つ新しい構造体（頂点構造体+2つ片辺構造体）をﾎﾟｲﾝﾀで連結する

HandleCircleEvent (γ)続き

3. αが消えることによって、ﾋﾞｰﾁﾗｲﾝは変化する。新たに形成された2組の3連続弧を調べる
   1. αの左弧を中央の弧とする3連続弧→もし2つのﾌﾞﾚｰｸﾎﾟｲﾝﾄが最終的に一点に集中する→対応する円ｲﾍﾞﾝﾄをQに挿入し、この新しい円ｲﾍﾞﾝﾄとTの対応する葉の間をﾎﾟｲﾝﾀで連結する
   2. αの右弧を中央の弧とする3連続弧について→ステップ3-1と同様な処理を行う

計算量

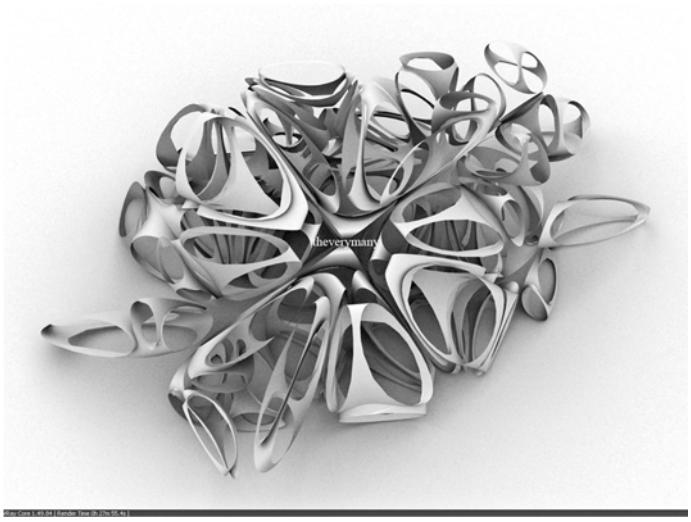


- 1つイベントを処理するのに必要な基本操作
  - 走査線状態木＋イベントキュー
    - 木要素の挿入･削除操作＝$O(\log n)$
  - イベント数＝$O(n)$
    - サイトイベント数＝n
    - 円イベント数(頂点数)≦$2n-5$(性質6)
    - サイトイベント+円イベント≦$3n-5 \rightarrow O(n)$
  - 全部の計算量→$O(n\log n)$

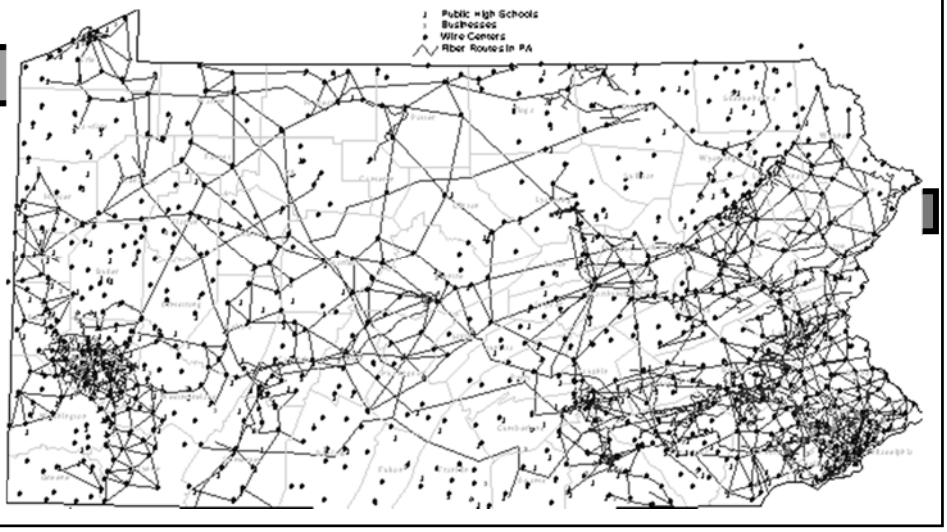
応用

- Chemistry, Physics, Biology, Medicine
- Mechanics, Hydrodynamics
- Ornamental design
- Forestry, Zoology
- Geography, Geology, Terrain modeling
- Route Planning
- Telecommunications network

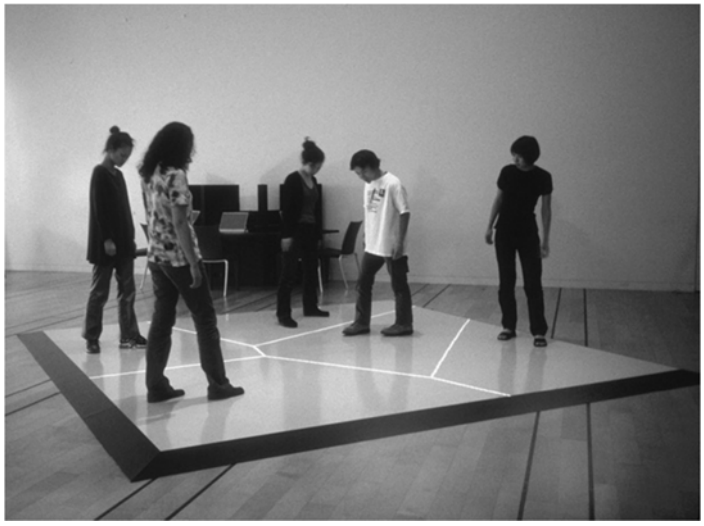
インテリア設計



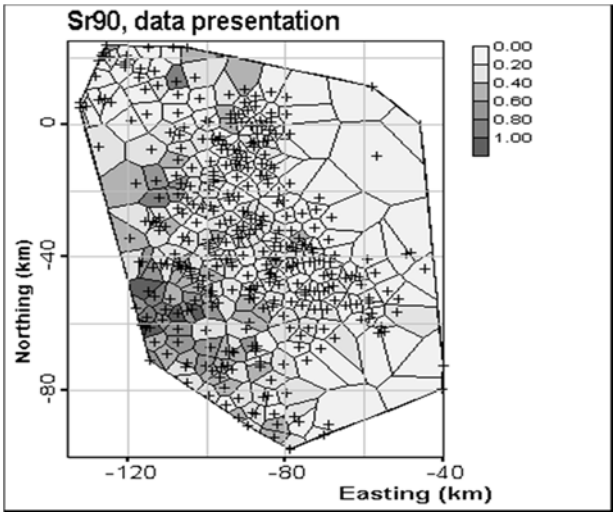
通信ネットワーク



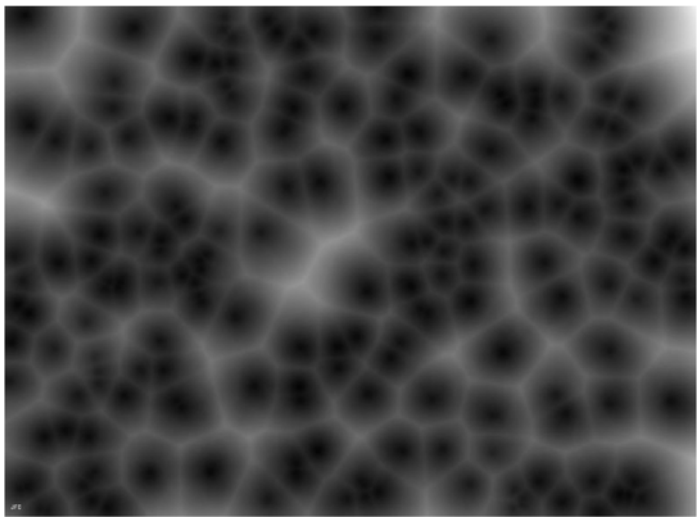
ボディゲーム



原子炉事故による土壌汚染



コンピュータアート



Visual Demonstration