



計算幾何学 Computational Geometry



第四章 ポロノイ図 Voronoi Diagram

ある会社の営業所分布

エリアマーケティングの展開。



資本金	26億3,498万円
取引先	約250社
お得意先軒数	
病院	約2,500軒
開業医	約29,500軒
薬局	約15,000軒
合計	約35,000軒

経営戦略

- 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

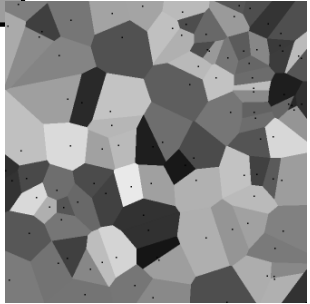
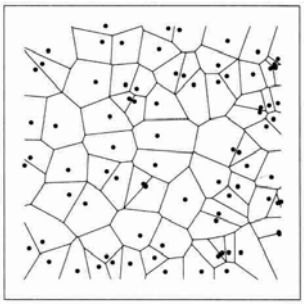
様々な最近点問題

- 公衆電話
- 郵便ポスト
- レストラン
- 電车站
- 高速IC
- バス停
- コンビニ



解答→ポロノイ図

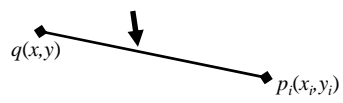
- 第一印象？

問題の一般化

- 平面上の n 個の点集

$$P = \{ p_i(x_i, y_i) \mid i=1, 2, \dots, n \}$$
 とする。
- 任意の点 $q(x, y)$ に対して、どの p_i が最も近い？
- 2点間のユークリッド距離 (Euclidean distance)

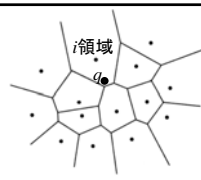


定義

- 平面上に n 個の異なる点の集合 P
 $P = \{ p_i(x_i, y_i) \mid i=1, 2, \dots, n \}$
- 点 $p_i \leftrightarrow$ 領域 $i \Rightarrow n$ 個の平面領域分割
- 領域 i に含まれる任意点 q について
 下記に関係が成立

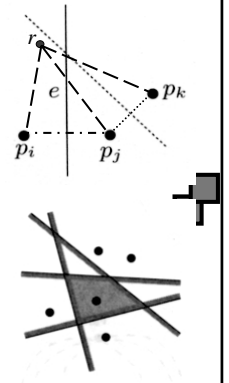
重要な関係式!

- 上記性質を持つ平面の領域分割
 $\Rightarrow P$ のボロノイ図



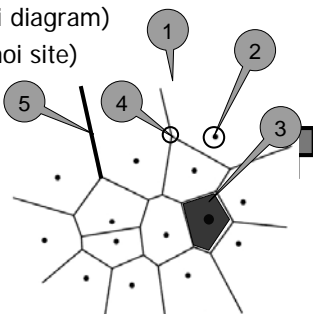
一つの領域の構造

- 2点間の線分の垂直2等分線
- 平面を2つの半平面に分割
 - $h(p_i, p_j) \rightarrow p_i$ を含む開半平面
 - $h(p_i, p_j) \rightarrow p_j$ 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)



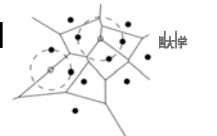
性質

1. ボロノイ領域 \rightarrow 凸領域
2. 有界でないボロノイ領域 \Leftrightarrow 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 \rightarrow 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)



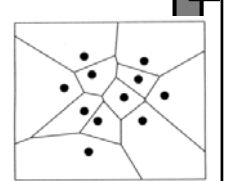
性質 続き

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 $\rightarrow O(n^3)$
 - 垂直2等分線の定義による方法
2. 逐次添加法 $\rightarrow O(n^2)$
 - 3点から始めて、1点ずつ添加していく
3. 分割統治法 $\rightarrow O(n \log n)$
 - 分割 \rightarrow 構成 \rightarrow 合成
4. Fortune走査法 $\rightarrow O(n \log n)$
 - 巧妙で高速な平面走査法



直接法

- 母点 p_i と p_j の垂直2等分線は平面を2つの半平面 $h(p_i, p_j)$ 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 の方が近いため、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—幾何的方法

2等分線の求め方—解析的方法

- $p_1 p_2$ 線分 $y = \alpha x + \beta$
 - $\alpha = (y_2 - y_1) / (x_2 - x_1)$
 - $\beta = y_1 - \alpha x_1 = (x_2 y_1 - x_1 y_2) / (x_2 - x_1)$
 - $\theta = \text{tg}^{-1} \alpha$
- 垂直2等分線 $y = \alpha' x + \beta'$
 - $\alpha' = \text{tg}(\theta + 90^\circ) = -\text{ctg} \theta = -1/\text{tg} \theta = -1/\alpha$
- 交差点で、 $y' = y, x = (x_1 + x_2) / 2$
 - $\alpha' x + \beta' = \alpha x + \beta$
 - $\beta' = (\alpha - \alpha') x + \beta = (\alpha - \alpha')(x_1 + x_2) / 2 + \beta$

一つのボロノイ領域

- 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 $V(p_k)$

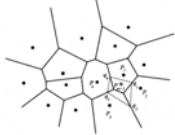
- 初期化
- for $i=1$ to n
- if $i \neq k$ then
- 線分 $p_i p_k$ の垂直2等分線を引く
- p_k 側の開半平面 $h(p_k, p_i)$ を求める
-

計算量

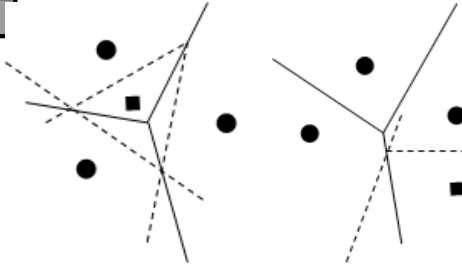
- 2本の2等分線の交差点を求める
 - $\rightarrow 1$ つの計算単位
 - $H(p_1, p_2) \cap H(p_1, p_3) = 1$ つの計算単位
 - $H(p_1, p_2) \cap H(p_1, p_3) \cap H(p_1, p_4) = 2$ つの計算単位
 - ...
 - $H(p_1, p_2) \cap H(p_1, p_3) \cap H(p_1, p_4) \dots \cap H(p_1, p_n) = n-2$ の計算単位
- 1個領域の全部計算量
 - $1 + 2 + \dots + (n-2) = (n-1)(n-2)/2 = O(n^2)$
- n 個領域の全部計算量 $= O(n^3)$

逐次添加法

- まずは3つの母点 p_1, p_2, p_3 からスタートして、ボロノイ図を構成する
- 残りの母点を一つ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

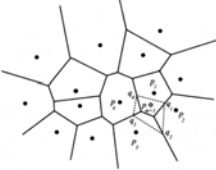


2つの例



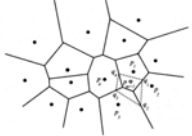
アルゴリズム

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, \dots, 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
4. 線分 $p_{(1)}p_{m+1}$ 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 $p_{(k)}p_{m+1}$ 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 ($i = 1, 2, \dots, k-1$) を順番に繋いで多角形 $q_1q_2 \dots q_{k-1}$ 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
13. } // end of for



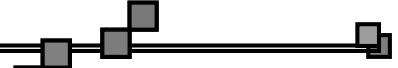
計算量

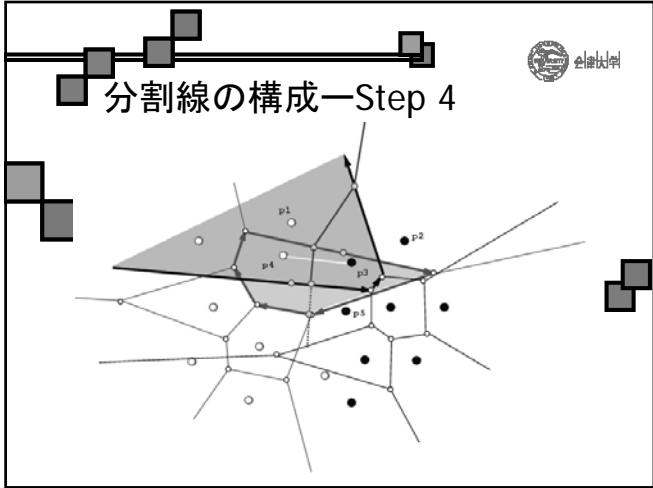
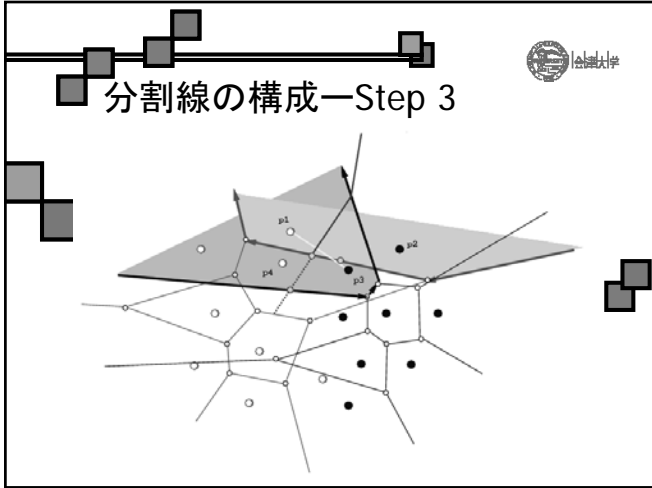
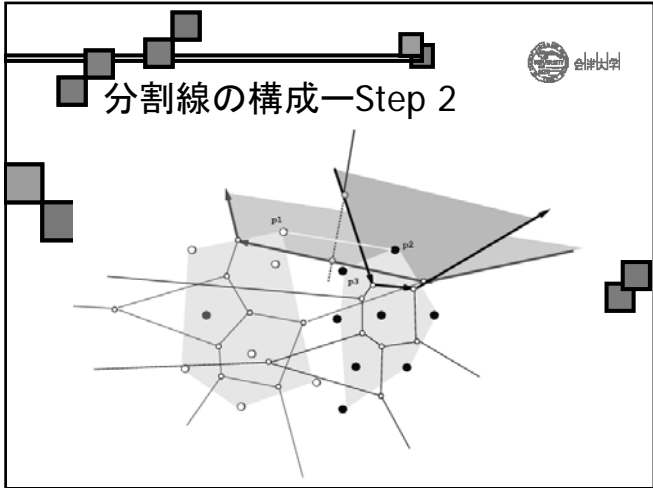
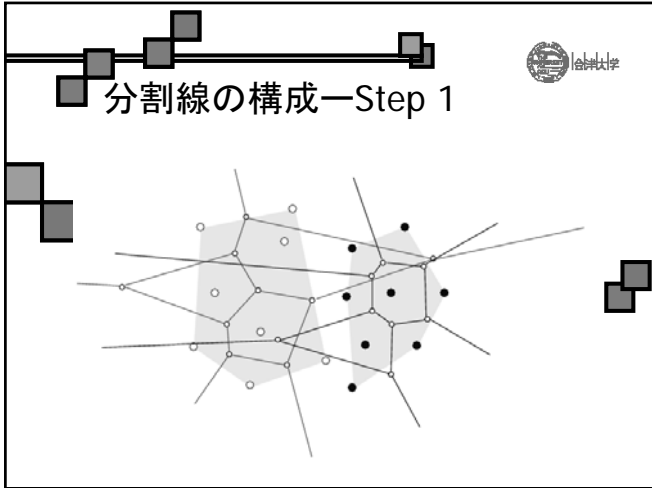
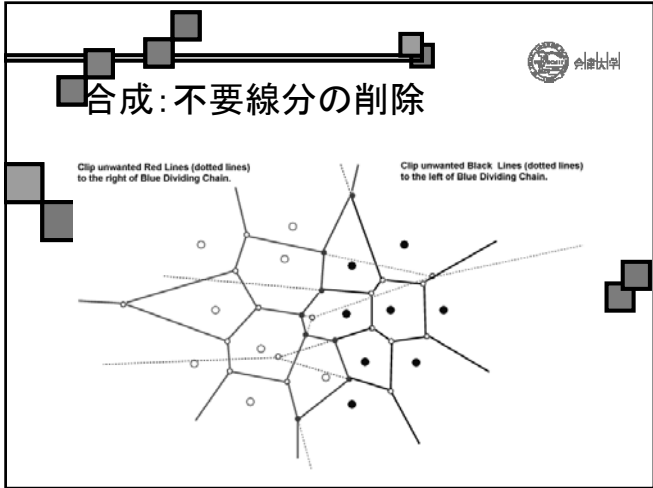
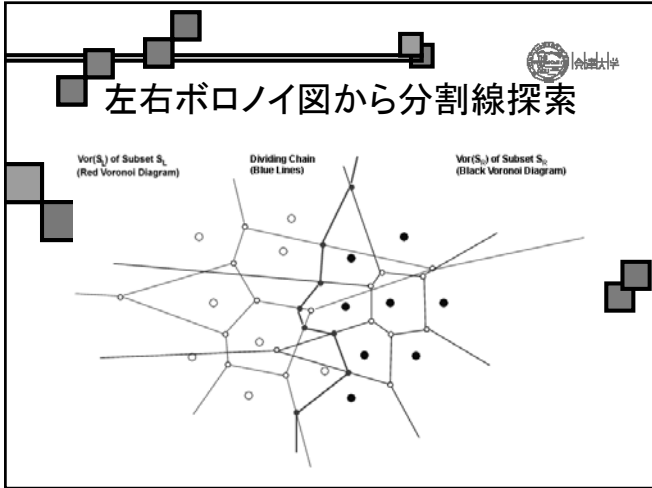
- 2点間の2等分線を求める
 - \rightarrow 1つの計算単位
 - 4点目 \rightarrow 3つの計算単位
 - 5点目 \rightarrow 4つの計算単位
 - ...
 - n点目 \rightarrow n-1の計算単位
- n点全部の計算量
 - $3+4+\dots+(n-1) = (n+2)(n-3)/2 = O(n^2)$

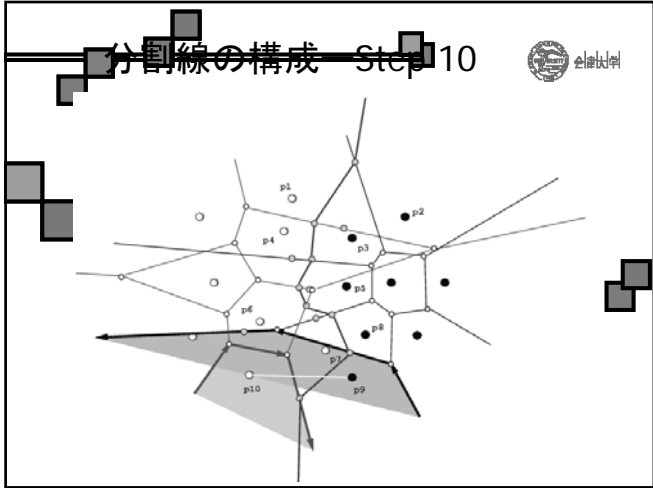
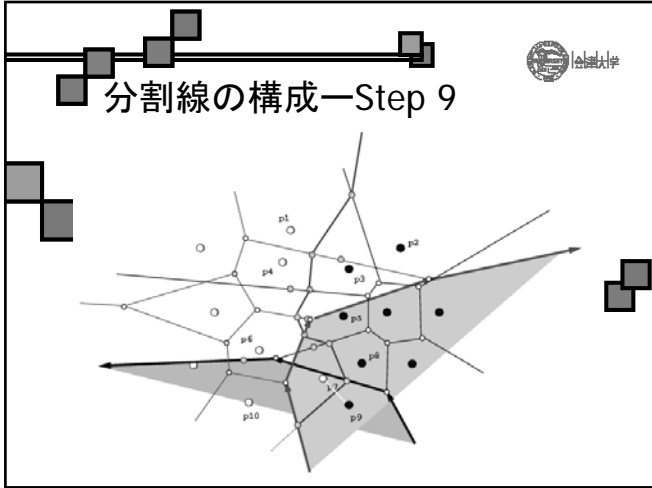
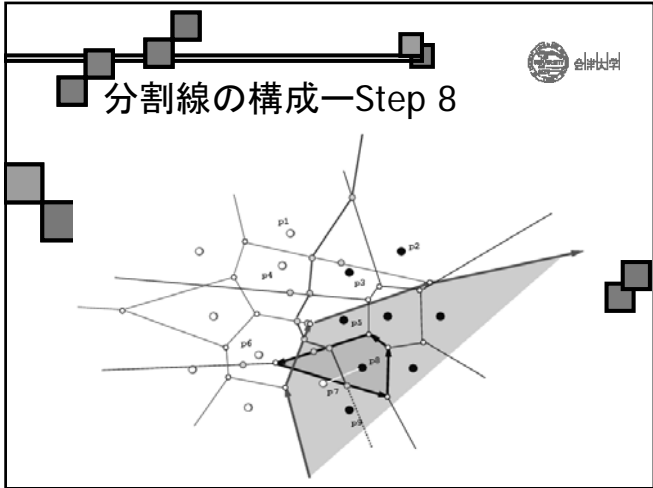
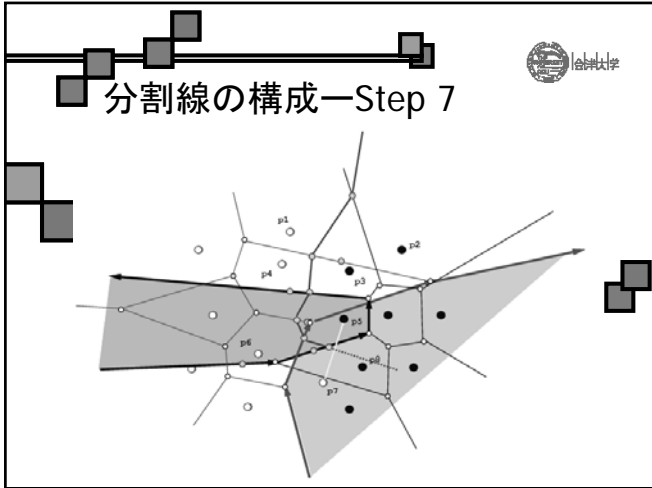
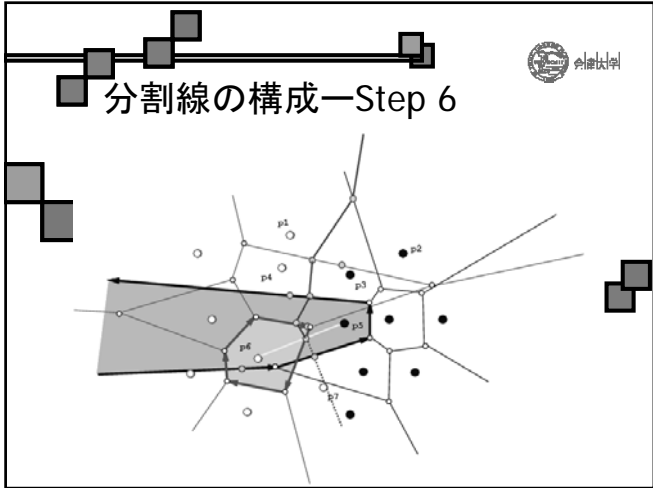
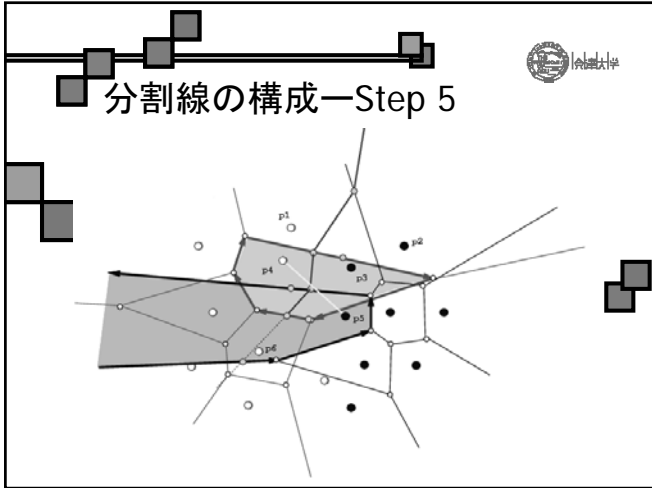


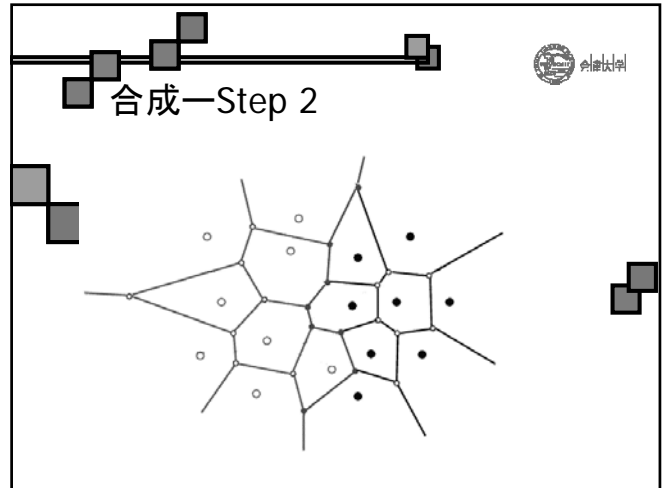
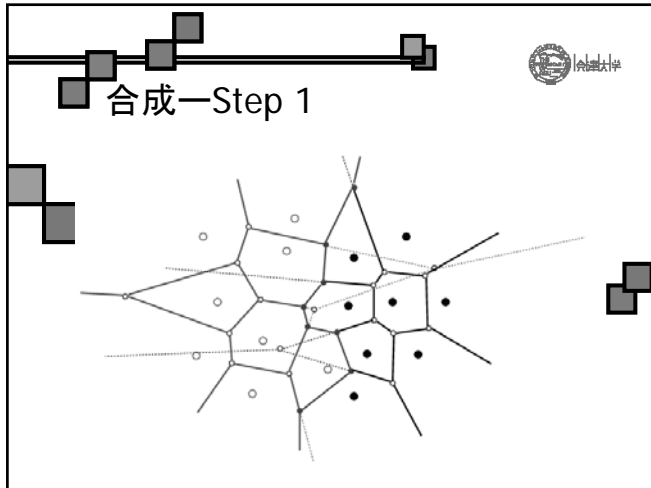
分割統治法

- The '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 the Voronoi diagram, the set of sites, S , is split up by a **dividing line** into subsets S_L and S_R of about same sizes. Then, the Voronoi diagram, $\text{Vor}(S_L)$, of subset S_L and Voronoi diagram, $\text{Vor}(S_R)$, of subset S_R are computed recursively.
- Shamos and Hoey, the first $O(n \log n)$ algorithm, 1975









■ 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 $P = \{p_1, p_2, \dots, p_n\}$, x 座標の昇順
- 出力: ポロノイ図 $Vor(P)$

1. $t = \text{integer}(n/2)$, let $P_L = \{p_1, p_2, \dots, p_t\}$, $P_R = \{p_{t+1}, p_{t+2}, \dots, p_n\}$
2. 帰帰的に P_L のポロノイ図 $Vor(P_L)$ を構成する
3. 帰帰的に P_R のポロノイ図 $Vor(P_R)$ を構成する
4. MERGE_VORONOI を用いて $Vor(P_L)$ と $Vor(P_R)$ を合成し、 $Vor(P)$ を構成する
5. $Vor(P)$ を返す

■ アルゴリズム(1/3)

重要なルーチン

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

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 $U(P_L, P_R) \neq$ lower common support line
 - ① $i \leftarrow i + 1$
 - ② $B(p_L, p_R)$ と $V(P_L)$ の境界線との交差点 a_L を探索する
 - ③ $B(p_L, p_R)$ と $V(P_R)$ 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
 - $w_i \leftarrow a_L$
 - $p_L \leftarrow a_L$ の所在するポロノイ辺の反対側にある母点
 - else
 - $w_i \leftarrow a_R$
 - $p_R \leftarrow a_R$ の所在するポ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. $w_{m+1} \leftarrow$ 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 $(w_0 w_1, w_1 w_2, \dots, w_m w_{m+1})$ を加える
9. $V(P_L)$ から polygonal line の右部分を削除する
10. $V(P_R)$ から polygonal line の左部分を削除する
11. $Vor(P_L)$ と $Vor(P_R)$ を合成したポロノイ図 $Vor(P)$ を返す

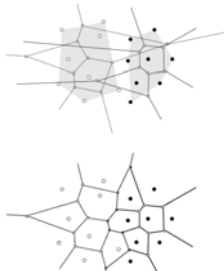
アルゴリズム

- 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: 頂点对 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. ①と②を交替に繰り返す
 - ① While 頂点 $next[u] > U(u, v)$, $u \leftarrow next[u]$
 - ② While 頂点 $next[v] > U(u, v)$, $v \leftarrow next[v]$
3. $U(u, v)$ を返す

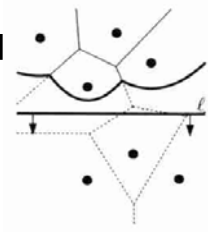
計算量

- UPPER_COMMON_SUPPORT
 - $O(n)$
- MERGE_VORONOI
 - 左右ボロノイ図にある関連母点の垂直二等分線の計算
 - $O(n)$
- VORONOI_DIAGRAM
 - 分割線の計算と左右ボロノイ図の合成
 - $T(n) = 2T(n/2) + O(n)$
 $= O(n \log n)$



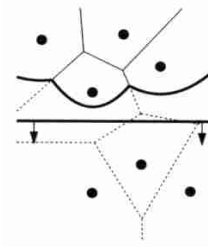
平面走査法?

- 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 を平面上で上から下まで動かしていきながら、ボロノイ辺の情報を求める?
- 結論: 大変 ← ボロノイ辺は走査線 l の上・下母点に依存するから!



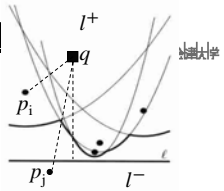
Fortune走査法 → 発想の転換

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 の下側に位置する母点によって変えられない(ボロノイ領域の性質 → 母点との距離)ように走査線 l の上側にある母点に関する部分的なボロノイ図の構造情報を管理する



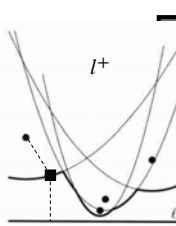
基本的な考え方


- $l^+ \leftarrow l$ の上側の閉半平面
- $l^- \leftarrow l$ の下側の開半平面
- 変わらない l^+ 上のボロノイ図の部分? ⇔ l^+ 上の他の母点より、ある母点 p_i に最も近い点 q の最大範囲が分かれば → ボロノイ領域 $V(p_i)$
- l^+ の点 $q \rightarrow \text{dist}(p_j \in l^-, q \in l^+) > \text{dist}(l, q \in l^+)$
- l^+ の点 $q \rightarrow \text{dist}(p_i \in l^+, q \in l^+) \leq \text{dist}(l, q \in l^+)$
 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)

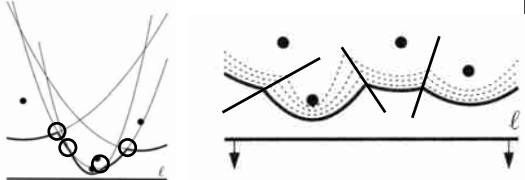
- 走査線 l までの距離と l^+ 平面の各母点との距離が等しい点は、それぞれの放物線になる
- これらの放物線の弧の系列 → beach line
- 母点 $p_i (p_{ix}, p_{iy})$ の放物線方程式
 $l_y \rightarrow$ 走査線の y 座標






ブレイクポイント(breakpoint)

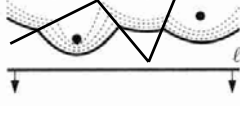
- ビーチラインを構成している各母点の放物線の弧の交点 → breakpoint
- 走査線が上から下まで動く間に、ブレイクポイントはポロノイ辺をたどっている！






考察

- 走査線を動かしながら、ビーチラインを管理すれば、ポロノイ辺が分る
- 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に、イベント点の時だけを考える
- ビーチラインの構造変化とイベント点の関係？

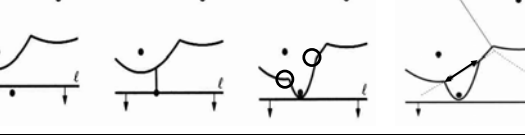





サイトイベント(site event)

- 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. 新たなブレイクポイントが2つ現れる(左図右)
3. 最初、2つのブレイクポイントは同一点に統合されるが、走査線の移動に伴って段々異なる方向に向かって新たなポロノイ辺をたどっていく(右図)

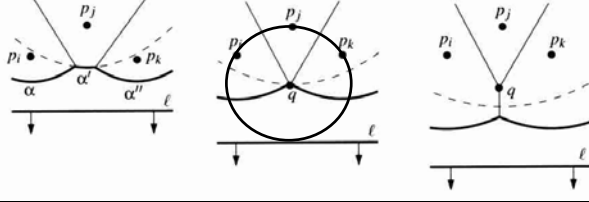





円イベント(circle event)

- 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 はポロノイ頂点 ← $(p_x - q_x)^2 + (p_y - q_y)^2 = R^2$






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 新たなポ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 2つの成長している辺が交差してポ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？

走査線を上から下まで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！ →

1. イベントのデータ構造
2. 走査線状態のデータ構造
3. ポロノイ図のデータ構造

イベントのデータ構造

- サイトイベント→事前分っている→y座標順でリストに保存
- 円イベント →事前分らない→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→ポロノイ頂点

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

円心と半径の計算方法

- 円の方程式 $(x - x_0)^2 + (y - y_0)^2 = R^2$
- 3つの未知数→既知3点があれば解ける

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

$$R = \frac{\sqrt{d^2 + e^2} \cdot f}{4a^2} \quad (x_2, y_2)$$

$$x_0 = \frac{-d}{2a} \quad y_0 = \frac{-e}{2a} \quad (x_3, y_3)$$

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

走査線状態のデータ構造

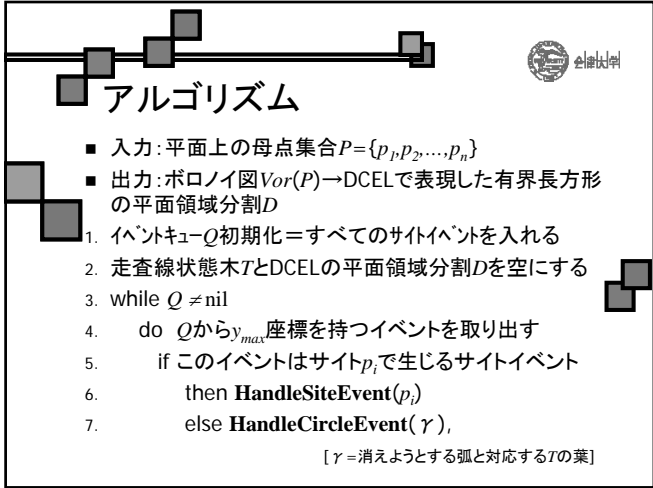
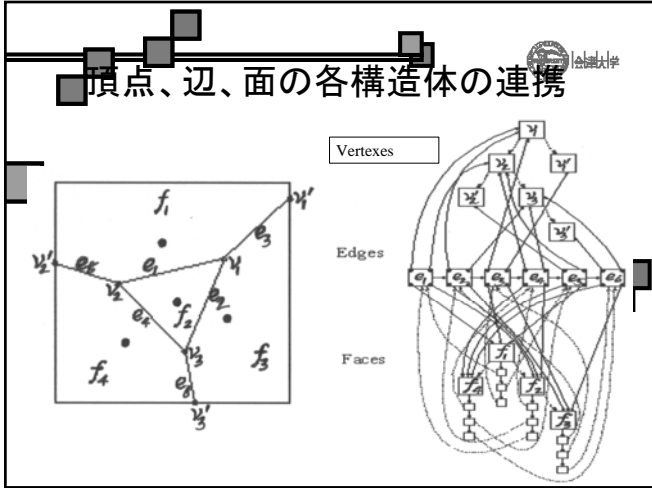
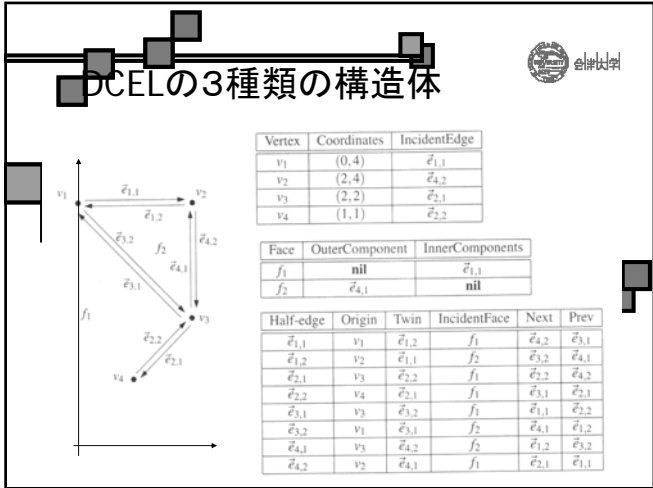
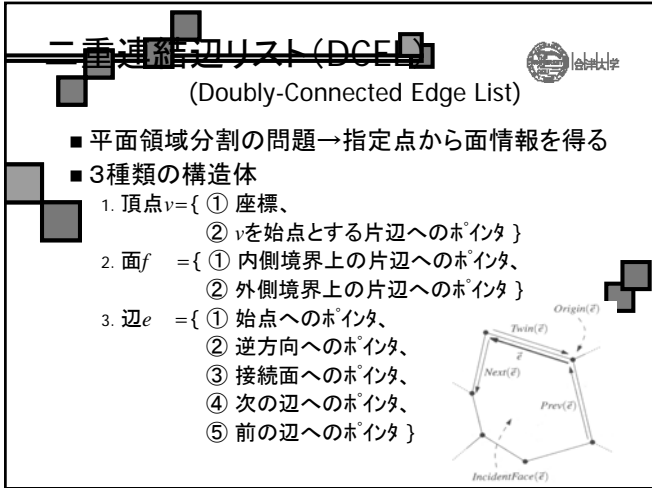
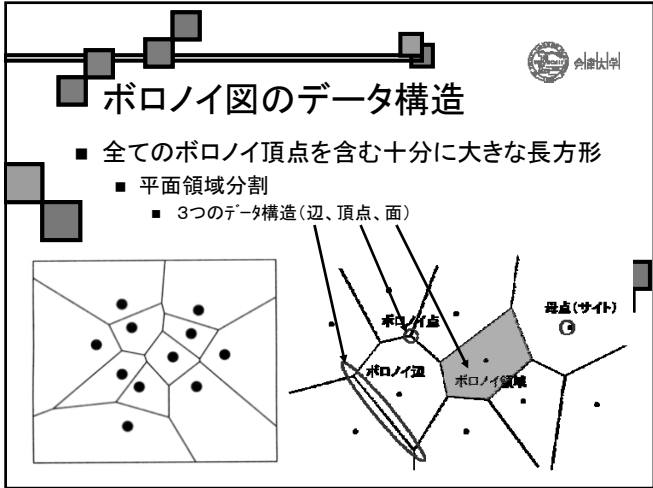
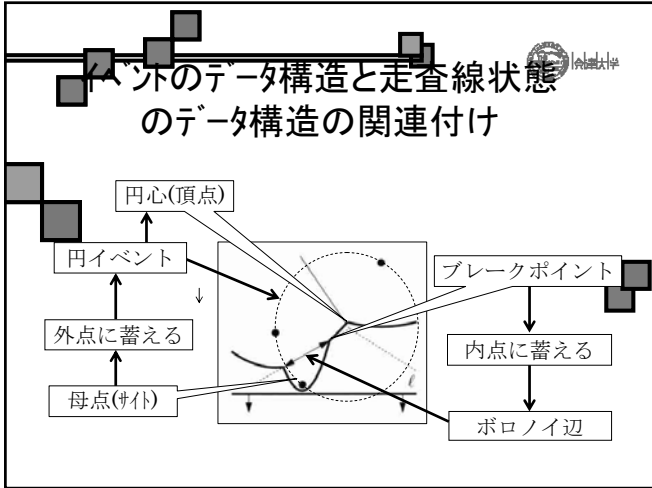
- ビーチライン→放物線の弧→明示に保存しない→母点とブレイクポイント→2分探索木 T
- T の外点→各弧に対応する母点 (x座標順)
- T の内点→各弧のブレイクポイント (母点の順序対 $\langle p_i, p_j \rangle$)
 - p_i = ブレイクポイントの左にある放物線を定義する母点
 - p_j = ブレイクポイントの右にある放物線を定義する母点



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が融合して、弧aと弧cによる新たなブレイクポイントを形成する





アルゴリズム 続き

- T 内にあるすべての内部節点は、ポロノイ図の片無限辺に対応している
- 既存すべての頂点を囲む有界長方形を求める
- 片無限辺を有界長方形に接続するように、DCELを更新する
- 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ポイントを追加する



HandleSiteEvent (p_i)

- If $T = \text{empty}$ then $\{p_i$ を T に挿入, $\text{return}\}$ else Do 2~5
- 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 中の円イベントへのポイントを持っていれば、この円イベントは余計なもので、 Q から削除
- 弧 α の対応する 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つの葉には、 p_i を蓄える(元々の α と対応する)。また、2つ新しい内点に2つの新たなブレイクポイント $\langle p_i, p_i \rangle$ と $\langle p_i, p_i \rangle$ を蓄える。必要なら、 T を平衡化する



HandleSiteEvent (p_i) 続き

- 新しい片辺構造体を生成し、 p_i と p_j のポロノイ領域を分割する
- 新たな弧を含んだ連続的となる三つの弧を調べ、2つの処理を行う
 - 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つのブレイ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 の節点の間のポイントを追加する。
 - 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


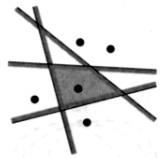
HandleCircleEvent (γ)

- 消えようとする弧 α と対応する葉 γ を T から削除する。対応する内点(ブレイクポイント)も更新する。必要なら、 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
- この円イベントの圆心を新しい頂点構造体として D に追加する。ヒッチラインの新しいブレイクポイントと対応する2つの片辺構造体を生成する。その間のポイントも追加する。この新しい頂点を終点とする片辺構造体に3つの新しい構造体(頂点構造体+2つの片辺構造体)をポイントで連結する

HandleCircleEvent (γ) 続き

- α が消えることによって、ヒッチラインは変化する。新たに形成された2つの3連続弧を調べる
 - α の左弧を中央の弧とする3連続弧→もし2つのブレイクポイントが最終的に一点に集中する→対応する円イベントを Q に挿入し、この新しい円イベントと T の対応する葉の間をポイントで連結する
 - 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 = $O(\log n)$
 - サイトイベント数 = n
 - 円イベント数(頂点数) $\leq 2n - 5$ (性質6)
 - $3n - 5 \rightarrow O(n)$
 - 全部の計算量 $\rightarrow O(n \log n)$

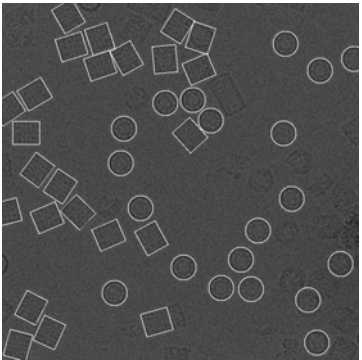
応用

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

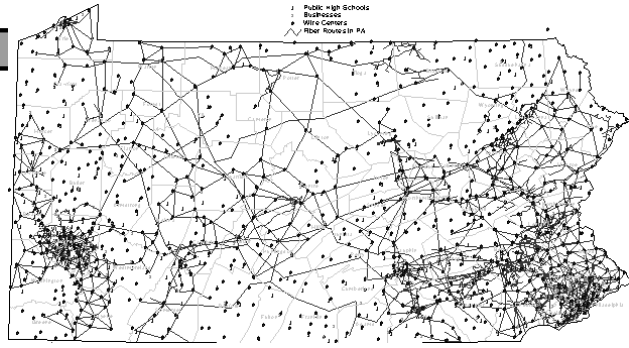
インテリア設計



電子顕微鏡画像処理



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

Sr90, data presentation

