□ 技 術 解 說 □

# 전산 기하학과 Voronoi도표

左 京 龍*

■ 차 례 ■



## 1 서 론

컴퓨터의 처리 능력이 방대해지고, 또한 이의 응용이 확대됨에따라 전산기하학 (Computational Geometry) 이 전산학의 한 중요한 분야로 등장하고 있다. 전산기하학이란 기하와 관련된 여러 문제들을 컴퓨터를 이용하여 해결해 보자는 의도에서 출발한 분야이다. 기하학이라면 적어도 2500년의 역사를 가진 학문으로서 다른 여느 학문에 비해 그 기틀이 잘 잡혀있다고 하겠다. 하지만 컴퓨터의 출현 훨씬 이전부터 이미 그 기틀이 잡혀있음으로 해서 종래의 기하학에서 다루던 아이디어나 많은 고전적인 결과들이 전산학에서 주로 다루는 알고리즈믹한 면에서 보면 여러가지 문제점을 많이 내포하고 있다. 어느문제를 컴퓨터로 푸는데 있어서는 단순히 알려져있는 공식들을 프로그래밍 언어로 바꾸어 쓴다고 해서 되는게 아니라 그 문제를 어떻게 나타내고, 데이타 구조를 어떻게 짤 것이며, 그 문제를 푸는 효율적인 알고리즘은 어떻게 설계를 하고, 또 그 Computational Complexity 는 어떤가 하는 등등의 중요한 문제점들을 고려해야 한다. 이러한 관점에서 볼 때, 특히 기하문제에 있어서는, 고전적인 결과들이 반드시 가장 좋은 알고리즘을 산출해내는 건 아닌 것이다. 그 이유는 종래의 기하학에서는 콤파스와 자 만이 기본적 도구로서 사용되었고 또 될 수록 간결한 정리들을 만들어 그 정리들이 얼마나 더 새로운 정리들을 증명할 수 있는가에 더 중점을 두어 왔기 때문인 것이다. 따라서 기하에 관한 여러 문제들을 좀 더 구체적이고도 효율적인 알고리즈믹한 형태로서 재현 하고자 하는 것이 바로 전산 기하학에서 추구하는 핵심이라 하겠다. 하나의 예를 들어 "평면상에 주어진 n 각형이 Convex 인가 아닌가" 를 판정하는 문제를 고려해 보자. 종래의 기하학에서 이미 잘 알려져 있는 기본 정리는 다음과 같다. 기본정리: 다각형 P가 Convex 할 필요 충분 조건은 다각형 P 의 모든 diagonal 이 P의 부분 집합이다.

위의 정리 자체가 하나의 완전하고도 좋은 특성을 말하고 있으나 이를 이용하려면 n각형이 모든 diagonal 의 수가 $\binom{n}{2}$개 만큼 있으므로 적어도 $O(n^2)$의 operation이 필요하다. 그러나 이 문제를 푸는데 있어서 설명은 복잡한 편이나 $O(n)$의 operation 으로 할 수 있는 알고리즘이있다.[13]

본 기술 해설에서는 전산 기하학에서 다루는 많은 기본 문제들 중에서도 특히 평면상에 놓여있는 n 개의 점들에 대한 여러 문제, 예를들면 Euclidean Minimum Spanning Tree를 구하는 문제, 점 사이의 거리가 가장 가까운 두 점 (two closest point pair ) 을 찾는 문제, Convex hull을 찾는 문제 등을 효율적으로 처리할 수 있는 Voronoi 도표 (Voronoi Diagram) 라는 기본적인 structure에 대해 설명을 하고 이 Voronoi 도표가 위에서 언급한 문제를 해결 하는데 이용됨을 살펴보고자 한다.

먼저 다음 절에서는 알고리즘을 분석하는데 필요한 개념들을 설명하겠다.

* 韓國科學技術院 電算學科 敎授 · 工博

## 2 알고리즘과 그 complexity

알고리즘을 평가 하는데 있어서 가장 널리 쓰이고 있는 기준 중의 하나는 문제의 instance가 점점 커짐에 따라 그 문제를 푸는 알고리즘의 수행 시간이 어떠한 패턴으로 증가 하는가를 알아보는 것이다. 어느 문제의 크기를 그 입력 데이타의 양으로 측정하여 정수로서 나타낸다 할때 — 예를들어, 평면 상에 주어진 점들 중 어떠한 특성을 갖는 집합을 찾는 문제라 하면 그 문제의 크기는 평면 상에 주어진 점의 수 — 그 알고리즘의 수행 시간은 문제의 크기의 함수로서 나타낼 수 있으며 이를 그 알고리즘의 "time complexity"라 부른다. 여기에서 시간의 개념을 좀 더 명확히 하기 위해서는 알고리즘을 수행하는 Computation모델을 밝힐 필요가 있다. Computation모델에는 Primitive한 Turing Machine 모델로 부터 실제 쓰이고 있는 어느 특정한 컴퓨터에 이르기 까지 다양하게 있겠지만 여기서는 real RAM (real random-access machine)[13] 을 사용하겠다. 그 이유는 Turing machine은 기초 이론적인 연구에 유용한 반면에 어느 특정 컴퓨터는 기종마다 특성이 다를뿐더러 그 해석이 필요 이상으로 어렵고 복잡하기 때문이다. real RAM (RAM에대한 자세한 내용은〔1〕참조)의 기본 operation은 사칙계산, 두실수의 비교, branching과 대수, 지수 및 삼각함수들과 같은 임의의 analytic function등이며 이들 각 operation을 수행하는데 공히 한 단위 시간이 걸린다고 가정한다.

알고리즘들을 비교하는데 있어 정확한 수행시간을 찾기 보다는 문제의 크기가 증가함에 따라 그 Complexity의 limiting behavior를 알아보는, 즉 asymptotic time complexity를 사용함이 편리하다. 따라서 이를 위한 표기법으로서 다음과 같은 정의를 사용한다.

"$f(n)=O(g(n))$ 이라는 것은 모든 $n \geqq n_0$ 에 대해 $|f(n)| \leqq c|g(n)|$ 을 만족하는 두 양의상수 $c$ 와 $n_0$ 가 존재한다는 것과 같다."

위의 정의에서 $f(n)$을 문제의 크기 $n$에 대한 어떤 알고리즘의 time complexity라 할때 우리는 그 상한선 $|g(n)|$을 찾아서 이를 $O(g(n))$이라 표기한다 앞으로 편의상 time complexity를 asymptotic time complexity로 간주 하겠다. 알고리즘의 time complexity도 두 종류로 나눌 수 있는데 주어진 문제의 크기에 대해 그 크기의 모든 가능한 입력들 중 가장 큰 complexity를 택하는 worst-case time complexity와 주어진 크기의 모든 가능한 입력들에 대해 어떠한 distribution (통상적으로 uniform distribution) 하에서 평균치의 complexity를 택하는 average-case time complexity가 있다.

이제 마지막으로 중요한 개념인 문제의 lower bound와 upper bound에 대해 설명하고자 한다. 어느 문제의 lower bound란 이 문제를 푸는 모든 가능한 알고리즘들 중 그 worst-case complexity가 가장 작은 값을 말하며, 또 upperbound란 현재까지 알려진 모든 알고리즘들 중 worst-case time complexity가 가장 작은 값을 말한다. 이를 다시 말하면, lower bound가 $L(n)$이란 것은 이 문제를 푸는 어떠한 알고리즘도(즉 누가 새로운 알고리즘을 개발한다 해도) worst-case에는 적어도 $L(n)$만큼의 시간을 요한다는 말이며, 또 upper bound가 $U(n)$이란 것은 $U(n)$만큼의 시간으로서 그 문제를 해결할 수 있는 알고리즘이 현존한다는 의미이다. 따라서 우리가 어느 문제의 새로운 upper bound를 얻었다는 말은 현재까지 가장 빠른 알고리즘을 개발했다는 말과 같으며, $U(n)=L(n)$이란 최적 알고리즘을 얻었다는 의미로서 바로 이것이 알고리즘 개발의 궁극적인 목표인 셈이다.

## 3 Voronoi도표와 그 특성

Voronoi 도표는 주어진 n개의 점에 대해, 평면의 전 영역을 n개의 소영역으로 분할한 것인데 이 각각의 소영역은 주어진 n개의 점 중 한 개의 점을 내부에 포함하고 있고, 이 소영역 내에 임의의 한 점을 잡았을 때, 이 점과 그 영역 내에 있는 주어진 점과의 거리가, 나머지 다른 주어진 어떤점과의 거리보다도 짧게 되도록 영역을 분할한 것이다.

이 분할된 각각의 소영역을 Voronoi영역이라 부른다.

이를 좀 더 수식적으로 나타내어 그 특성들을 살펴 보고자 한다.

$P=\{p_1, p_2, \cdots, p_n\}$를 평면상에 주어진 n개의 점들의 집합이라 하자. 또한 각 점은 그들의 $x$, $y$ 축의 좌표 $p_i=(a_i, b_i)$로서 나타나있다 하자. 각 점 $p_i$에 대해서 Voronoi 영역 $R_i$는 다음 식과 같이 표현된다.

$$R_i = \{x \in E_2 \mid d(p_i, x) \leq \min_{p_j \in P} \{d(p_j, x\}\}$$

여기에서 $d(x, y)$는 평면상의 임의의 두 점 $x$, $y$

간의 Euclidean 거리를 말한다.

잠시 주어진 임의의 두 점 $p_i$, $p_j \in P$를 고려해 보자 $p_i$ 와 $p_j$ 에서 같은 거리에 있는 점의 궤적은 바로 $p_i$ 와 $p_j$ 의 수직 이등분선 $h(p_i, p_j)$ 가 된다. 더우기 $R_{ij} = \{x \in E_2 \mid d(p_i, x) \leq d(p_j, x)\}$ 라 두면 $R_{ij}$는 수직 이등분선 $h(p_i, p_j)$ 로 나누어진 반 평면 중 $p_i$ 를 포함하고 있는 반 평면임을 나타내고 이를 이용하여 $R_i = \bigcap_{j \neq i} R_{ij}$ 를 얻으며 이는 기껏해야 $(n-1)$ 각형을 넘지 않는 볼록 다각형 영역(Convex polygonal region)이 됨을 알수 있다. 또한 Voronoi 영역의 경계선들을 합한(union), 즉 $\bigcup_{i \neq j} (R_i \cap R_j)$로 나타난다. 그림 1은 Voronoi 도표의 한 예를 보인 것 이다. 그림과 같이 Voronoi 도표에서의 꼭지점을 Voronoi 점이라 부르고 또 그 선분을 Voronoi edge 라 부른다.

평면상에 주어진 n 개의 점들로부터 이러한 Voronoi 도표를 구하는 알고리즘은 그 Voronoi도표를 나타낼 수 있도록 다음과 같은 데이타를 얻을 수 있어야 하겠다.

i) 각 Voronoi 점들의 좌표
ii) 각 Voronoi 점에 인접한 Voronoi edge 들 (두 Voronoi 점으로서 표시된다)
iii) 각 Voronoi edge 를 결정하는 원래 주어진 두 점
iv) 각 Voronoi 다각형의 edge 들을 cyclic order 로 나타낸 것

이러한 Voronoi 도표를 구하는 알고리즘은 많은 사람들에 의해 제시된 바 있는데[15] 70년대 중반까지만 해도 그 중 제일 효율적이라는 것이 $O(n^2)$ 알고리즘이었다. 그 이후 Shamos[13]에 의해 그 Lower bound 가 $\Omega(nlogn)$임을 보여주었으며 또한 실제로 $O(nlogn)$에 Voronoi 도표를 구할 수 있는 알고리즘을 제시하였다. 그 알고리즘의 기본 idea는 Divide and Conguer 방법, 즉 주어진 $n$ 개의 점들을 $x$ 좌표의 크기 순으로 반씩 나누어서 각각의 Voronoi 도표들을 recursively 구한 다음, 그 두 Voronoi 도표들을 하나의 도표로 linear time에 Merge 하는 것이다.

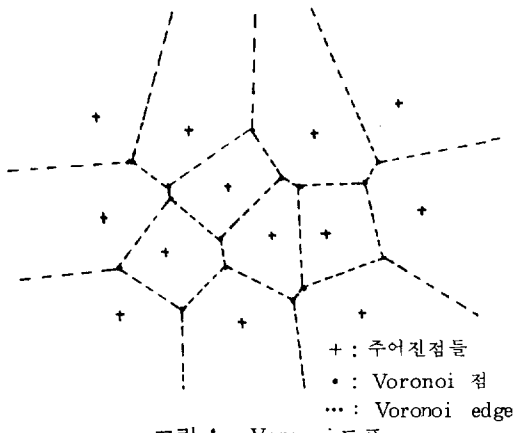

그림 1. Voronoi 도표

이제 다음 절에서의 설명을 위해 Voronoi 도표의 몇 몇 중요한 특성들을 증명없이 나열해 보고자 한다. 편의상 원래 주어진 $n$개의 점 중에서 어떠한 세 점도 한 직선상에 있지 않고 또 어떠한 네 점도 한 원주상에 있지 않다고 가정한다. (이러한 가정은 없다고 하더라도 조금만 보완함으로써 앞으로 논할 여러 명제들에 영향을 미치지 않는다).

1. 각 Voronoi 영역은 Convex 하다.

2. 어느 주어진 점 $p_i$ 의 가장 가까운 모든 이웃은 Voronoi 다각형 $V(i)$ 를 이루는 각 edge 에대해 $p_i$ 에 대응하는 점들 이다.

3. Voronoi 도표에서 bound 되지 않은 영역의 수는 주어진 점의 집합의 convex hull 을 이루는 점의 수와 같다.

4. Voronoi 점의 수와 Voronoi edge 의 수는각각 $2(n-1)-S$와 $3(n-1)-S$ 이다. 여기에서 $S$는 Voronoi 도표의 bound 되지 않은 영역의 수를 말한다.

5. Voronoi 도표의 직선 dual 은 triangulation 이다. (이를 Delaunay triangulation 이라 부른다)

6. Voronoi 도표에서 한 Voronoi 다각형의 edge 수의 평균치는 6을 넘지 않는다.

7. 각 Voronoi 점은 Delaunay triangle 의 외심 이다.

## 4 Voronoi도표와 그 응용

Voronoi 도표를 직, 간접으로 이용하는 문제는 여러 분야에 걸쳐 상당히 많으나, 이절에서는 Voronoi 도표를 하나의 기본 구조로 하여 이로 부터 좀 더 효율적으로 얻을 수 있는 7개의 기본적인 문제들에 관해 개략적인 알고리즘 을 설명 하고자 한다. 여기에서 주목할 것은 이 7개의 기본 문제들 중 G를 제외하고는 모두 그 lower bound 가 $\Omega(nlogn)$ 이다.*

### A. all nearest neighbors

문제 : 평면상에 주어진 $n$개의 점들로 부터 각각 가장 가까운 이웃 쌍 들을 찾아라.

하나의 흥미있는 현상은 위 문제보다 쉬운듯이 보이는 가장 가까운 쌍들 중에서도 제일 가까운 쌍 하나만 찾는 문제 (closet pair problem) 도 역시 lower bound 가 $\Omega(nlogn)$ 이다. 위 문제는 일단 Voronoi 도표만 찾으면 각 주어진 점 $p_i$ 의 가장 가까운 이웃은 Voronoi 다각형 $V(i)$ 의 edge 중에 하나 (특성 2) 이며, 또한 Voronoi edge 수가 $3n-6$ 넘지 않으므로 (특성 4) linear 시간에 구할 수가있다. 따라서 앞절에서 언급한 바와 같이 Voronoi 도표를 $O(nlogn)$ 에 구할수 있으므로 쉽게 최적 알고리즘임을 알 수 있다.

### B. Triangulation

문제 : 평면상에 주어진 $n$개의 점들로 부터 서로의 점들 간에 직선으로 연결하되 이들 직선의수가 최대가되고 또한 그들 주어진 점들 외에는 서로 교차하지 않도록 하라.

위 문제에서 직선으로 연결되는 edge들은 가능한 한 많은 삼각형을 형성하고, 주어진 n개의 점들로 이루어진 Convex hull 내에 있다. 따라서 triangulatin 은 planar graph 를 형성하므로 기껏해야 $3n-6$개의 edge 들을 가지고 있다. Voronoi 도표의 직선 dual을 구하면 linear 시간에 바로 triangnlation 이 되므로(특성 5) $O(nlogn)$의 최적 알고리즘을 쉽게 구할수 있다. 특히 triangulation 은 상당히 많은 종류가 있을수 있지만 위와 같이 구한 Delaunay triangulation 은 여러가지 좋은 특성(예를 들면, 대체로 가늘고 긴 삼각형은 잘 생기지 않는 경향)들을 가지고 있어서 여러 응용에 많이 쓰이고 있다.

### C. Euclidean Minimum Spanning Trce(EMST)

문제 : 평면상에 주어진 n 개의 점이 있을때 이들을 vertex로 하여 그 edge 길이의 합이 최소가 되는 tree 를 찾아라.

EMST 를 풀기 위해서 graph문제에서 잘 알려진 Prim 이나 Kruskal 의 알고리즘을 직접 사용 한다면 적어도 $O(n^2)$ 의 시간이 필요하다. 그러나 EMST 는 Voronoi 도표의 직선 dual 인 graph — 즉 Delaunay Triangulation — 의 subgraph [16] 라는 결과를 이용하면, 일단 Voronoi 도표를 찾고 난뒤, 그 dual 을 구하고, Kruskal 의 알고리즘 또는 planar graph 에서 minimum spanning tree 를 찾는 Cheriton [4] 의 $O(n)$ 알고리즘을 사용한다면 최적인 $O(nlogn)$ 시간에 찾을 수 있다.

### D. Gabriel graph

평면상에 주어진 $n$개의 점들로 구성된 집합 $P=\{p_1, p_2, \cdots\cdots, p_n\}$ 이 있다하자. $DISK(p_i, p_j)=\{x \quad E_2 \mid \{d^2(x, p_i)+d^2(x, p_j)\}^{1/2} \leqq d(p_i, p_j)\}$ 라 정의할때 주어진 점들의 쌍 $p_i$, $p_j$에 대해 만약 다른 모든 점들이 $DISK(p_i, p_j)$의 밖에 놓여 있다면 $p_i$ 와 $p_j$를 연결하여 형성된 gragh를 찾아라.

위의 문제에서 형성된 gragh를 Gabriel graph 라 하는데, 이 문제를 푸는 알고리즘 역시 "Gabriel graph는 Delaunay triangulation의 Supgraph" 라는 결과 [15] 를 이용하여 Voronoi 도표로부터 $O(n)$시간에 구할 수 있으므로 최적 알고리즘이 존재한다.

### E. Convex hull

문제 : 평면상의 n개의 점들이 주어졌을때 이 모든 점들을 포함하는 집합 중에서 최소의 면적을 가지는 convex한 집합을 찾아라.

위 문제에 대해서는 Graham [7], Shamos [13] 등 많은 사람들에 의해 이미 최적 알고리즘이 개발되어 있으나, 역시 voronoi 도표로부터도 특성 3과 5에 의해 linear time에 찾을 수 있다. 여기에서 특기할만한 사항은 convex hull 을 찾는 문제를 Expected case에는 $O(n)$에 풀수 있는 알고리즘도 있다.[3]

### F. Largest Empty Circle

문제 : 평면상에 주어진 n개의 점들이 있을때, 그 원 속에 주어진 점을 하나도 포함하지 않고 또 그 원의 중심이 convex hull 의 내부에 있는 모든 가능한 원 중에서 가장 큰 원을 찾아라.

n개의 주어진 facility 들이 있을때 이들 facility 들 중 어느 것으로부터든 가장 멀리 떨어진 새로운 한 facility의 위치를 찾는 문제가 바로 Largest empty circle의 중심을 찾는 문제와 동일하다. 이 문제는 75년도 까지만 해도 $O(n^3)$이 upper bound였으나, 이 원의 중심은 Voronoi 도표에서 Voronoi 점이거나 또는 Voronoi edge와 hull edge 와의 교차점 중에서 찾을 수 있다. 따라서 모든 Voronoi 점의수와 교차점 수가 $O(n)$이므로 최적인 $O(n \log n)$ 알고리즘을 찾을 수 있다.

---

* $f(n)=\Omega(g(n))$이라는 것은 모든 $n \geqq n_0$ 에 대해 $|f(n)| \geq c|g(n)|$을 만족하는 두 양의상수 $c$와 $n_0$가 존재 한다는 것과 같다.

**G. Smallest Enclosing Circle**

문제 : 편면상에 $n$개의 점들이 주어졌을때 이들을 모두 포함하는 원중 가장 작은 원을 찾아라

위 문제는 고전적인 문제 중의 하나로서 110여년 전부터 효율적인 알고리즘을 찾는 연구가 계속되어 왔지만 70년대 중반 까지도 $O(n^2)$가 upper bound였다.

앞의 Largest empty circle 문제가 maximin의 문제라면 Smallest enclosing circle 문제는 minimax 문제라 볼 수 있겠다. 따라서 지금까지 사용한 Voronoi 도표를 구하는 것과 유사한 방법으로 평면의 전 영역을 소 영역으로 분할하는데 이 각 각의 소 영역 $V(i)$는 그 영역내에 임의의 한점을 잡았을때 이 점과 점 $Pi$의 거리가 나머지 다른 주어진 어떤 점과의 거리 보다도 멀게 되도록 분할 할 수 있다. (이를 Farthest Voronoi 도표라 한다.) 이렇게 구성된 Farthest Voronoi 영역은 unbounded 된 영역들만 가지고 있으며 smallest enclosing circle의 중심은 이 도표의 Voronoi점들 중의 하나이거나 또는 주어진 점들의 diameter이다. 위의 사실을 이용하면 쉽게 $O(n \log n)$알고리즘을 찾을 수 있으며 이것이 현재까지의 upperbound 이다.

## 5 결 론

지금까지 우리는 전산 기하학에서 많이 다루는 기본 structure 중의 하나인 Voronoi 도표를 소개하고 이 Voronoi 도표가 기하의 여러 다른 문제들을 효율적으로 처리하는데 이용됨을 설명하였다.

끝으로 Voronoi 도표를 아래와 같이 일반화 내지는 확장시키는 방향의 연구도 진행되고 있다.

첫째, 본 기술해설에서는 모든 예들을 Euclideam 거리로서 다루었지만 $L_1$ 또는 $L_\infty$ norm에 이르기 까지 일반화 해서 확장하는 방향

둘째, 점에 대한 Voronoi 도표 뿐만 아니라, 선이라던지 또는 다각형, 원에 이르기 까지의 다양한 모형에 대한 Voronoi 도표를 구성하는 방향

세째, 평면만이 아니라 일반적인 $k$-dimension까지 확장하는 방향

네째, Voronoi 도표가 각 점들에 대한 것 뿐만 아니라 주어진 점들의 $k$-subset에 대한 Voronoi 도표(이를 $k$차 Voronoi도표라 한다)로의 확장 등이다.

## 참 고 문 헌


1) V. A. Aho, J. E. Hopcroft and J. D. Ullman; The Design and Analysis of Computer Algorithms. Addison-Wesley, 1974

2) N. Ahuja; "Dot pattern processing using Voronoi neighborhoods," IEEE on Pattern Analysis and Machine Intelligence, vol. PAMI-4, No.3, pp. 336—343, May 1982

3) J. L. Bently and M. I. Shamos; "Divide and conquer for linear expected time," Information Processing Letters, vol. 7, pp. 87~91, Feb. 1978

4) D. Cheriton and R. E. Tarjan; "Finding minimum spanning trees," SIAM J. of Computing, vol. 5, pp.724—742, Dec. 1976

5) J. Elzinga and D.W. Hearn; "Geometrical solutions for some minimax Location problems," Transportation Science 6, pp.379—394, 1972

6) R. L. Francis and J. A. White; Facility Layout and Location: An Analytical Approach, Prentice-Hall 1974

7) R. Graham; "An efficient algorithm for determining the convex hull of a planar set," Information Processing Letters, vol. 1, pp.132—133, 1972

8) P. J. Green and R. Sibson; "Computing Dirichlet tessellations in the plane," The Computer Journal, vol. 21, pp.168—173, 1978

9) R. A. Jarvis; "On the indentification of the convex hull of a finite set of points in the plane," Information Processing Letters, vol. 2, pp.18—21, 1973

10) D. T. Lee and R. L. Drysdale; "Generalization of Voronoi diagrams in the plane," SIAM J. on computing, vol. 10, pp.73—87, 1981

11) D. T. Lee and C. K. Wong; "Voronoi diagrams in $L_1(L_\infty)$ metrics with 2-dimensional storage applications," SIAM J. on Computing, vol. 9, pp.200—211, 1980

12) F. Preparata S. Hong; "Convex hulls of finite sets points in two and three dimensions," CACM, vol. 20, pp.87—93, 1977

13) M. I. Shamos; "Computational geometry," Ph.

D. thesis, Yale Univ., May 1978

14) M. I. Shamos and D. Hoey; "Closest-point problems," 16th Annual IEEE Symp. on Foundation of Computer Sicence, pp. 151—162, Oct. 1975

15) G. T. Toussaint ; "Pattern recognition and geometrical complexity," IEEE, 5th Int. Conf. on Pattern Recognition, pp.1324—1347, 1980

16) A. C. Yao; "An O(IEI log log | V | algorithm for finding Information Processing Letters, vol. 4, pp.21—23, 1975