$L_\infty(L_1)$ 디루니 삼각분할의 병렬처리 알고리즘

(A Parallel Algorithm for Constructing the Delaunay Triangulation in the $L_\infty(L_1)$ Metric)

위 영 철 †

(Young-Cheul Wee)

요 약 본 논문은 영역별 근접 그래프 (geographic nearest neighbor graph)와 레인지 트리 (range tree)를 이용하여 평면 위의 n 개의 점에 대한 $L_\infty(L_1)$ 거리 (metric) 상의 디루니 삼각분할 (Delaunay triangulation)을 구축하는 방법을 소개한다. 이 방법은 $L_\infty(L_1)$ 거리 상에서 디루니 삼각분할에 있는 각 삼각형의 최소한 한 선분이 영역별 근접 그래프에 포함됨을 이용하여 레인지 트리 방법으로 디루니 삼각분할을 구축한다. 본 방법은 $O(n\log n)$의 순차계산 시간에 $L_\infty(L_1)$ 디루니 삼각분할을 구축하며, CREW-PRAM (*Concurrent Read Exclusive Write Parallel Random Access Machine*)에서 $O(n)$의 프로세서로 $O(\log n)$의 병렬처리 시간에 $L_\infty(L_1)$ 디루니 삼각분할을 구축한다. 또한, 이 방법은 직선간의 교차점 계산 대신 거리비교를 하기 때문에 수치오차가 적고 구현이 용이하다.

Abstract *Using the geographic nearest neighbor approach, we introduce a new method for constructing the Delaunay triangulation for a set S of n sites in the plane under the $L_\infty(L_1)$ metric. Although there is no inclusion relationship between the geographic nearest neighbor graph and the Delaunay triangulation, we find that at least one edge of each triangle in the Delaunay triangulation is contained in a geographic nearest neighbor graph. Using this observation and employing the range tree scheme, we present an algorithm that constructs the Delaunay triangulation of S in $O(n\log n)$ sequential time. This algorithm can easily be parallelized, and takes $O(\log n)$ time with $O(n)$ processors on a CREW-PRAM (Concurrent Read Exclusive Write Parallel Random Access Machine). Because the operations involved in this algorithm are mainly distance comparisons rather than line intersections, this algorithm is numerically stable and easy to implement.*

1. 서 론

$L_\infty(L_1)$ 거리 (metric) 상의 보로노이 다이아그램 (Voronoi diagram: VD)은 로보틱스 (robotics), VLSI 설계, scheduling 문제 [9,10] 등 많은 분야에 활용되고 있다. 평면 위의 n 개의 점 S에 대한 VD는 평면을 S의 각 점에 근접한 영역으로 분할한 것이다 (그림 1 참조). S에 대한 디루니 삼각분할(Delaunay triangulation: DT)은 두 점 $a, b \in S$의 근접영역이 서로 인접하면 두 점 a, b를 에지로 연결한 그래프이다 (그림 2 참조). DT는 VD의 듀얼(dual) 그래프이므로 DT와 VD 구축은 같은 문제로 취급된다. VD를 구축하는 방법으로는 divide-and-conquer, sweep line, incremental, growing triangles, and lifting the sites into three dimensions 등이 있다 [12]. 평면 위의 n 개의 점에 대한 VD를 $O(n\log n)$의 최적시간에 구축하는 순차적 알고리즘들은 [12] 알려져 있으나, 병렬처리 알고리즘들은 CREW-PRAM (Concurrent Read Exclusive Write Programmable Random Access Machine)에서 $O(n)$ 프로세서로 $O(\log^2 n)$ 시간 [2,3] 또는 $O(n^3)$ 프로세서로 $O(\log n)$ 시간이 소요된다 [11]. Yao [16]에

· 본 연구는 아주대학교 "디루니 트라이앵귤레이션 알고리즘 개발" 과제의 지원을 받아 작성된 것입니다.

† 종신회원 : 아주대학교 정보 및 컴퓨터공학부 교수
ycwee@madang.ajou.ac.kr

논문접수 : 2000년 7월 12일
심사완료 : 2000년 12월 14일

의하여 소개된 영역별 근접 그래프 (geographic nearest neighbor graph: GNNG)는 minimum spanning tree (MST) [16,6], relative neighborhood graph (RNG) [7,13], shortest path motion planning [4], rectilinear Steiner tree [14] 등 여러 가지 근접성 문제에 활용되었다. 본 논문에서는 레인지 트리 (range tree) 방법을 도입하여 영역별 근접 그래프 (geographic nearest neighbor graph: GNNG)로 부터 $L_\infty(L_1)$ DT (따라서, VD)를 구축하는 방법을 소개한다. 이 방법은 $O(n\log n)$의 최적 순차계산 시간에 DT를 구축하며, CREW-PRAM 상에서 $O(n)$의 프로세서로 $O(\log n)$의 최적 병렬처리 시간에 DT를 구축한다.

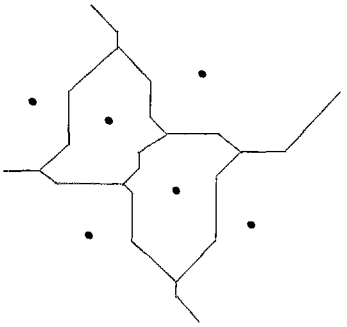

그림 1 L_∞ Voronoi diagram

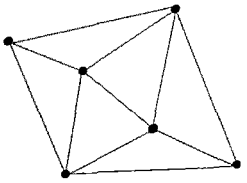

그림 2 L_∞ Delaunay triangulation

평면 위의 n 개의 점 S에 대한 한 점 a의 k 분면 근접 점은 점 a를 중심으로 k 개의 크기가 같은 섹터 (sector)로 분할한 각 영역에 대한 a의 근접 점이다 (그림 3 참조). 여기서 평면 분할은 y 축의 양의 방향 y^+이 항상 k 개의 분할선 중에 포함되도록 한다. S에 대한 영역별 근접 그래프 $GNNG_k(S)$는 S의 각 점에 대한 k 분면 근접 점들을 에지 (edge)로 연결 한 그래프이다. $QNG(S)$ (quadrant neighbor graph of S)는 S에 대한 사분면 근접 그래프 $GNNG_4(S)$를 나타내기로 한다 (그림 4 참조).

$MST(S)$, $RNG(S)$ 구축의 경우 $MST(S)\subseteq RNG(S)\subseteq GNNG_8(S)$의 관계를 이용하여 먼저 팔분면 근접 그래프 $GNNG_8(S)$를 구축 한 다음 이로부터 $MST(S)$, $RNG(S)$를 효율적으로 구축 할 수 있다 [16,6,7,13]. 그러나, $DT(S)$는 $GNNG_8(S)$의 부분집합이 아니므로 이 방법을 직접 적용 할 수 없다. 본 논문에서는, $DT(S)$에 있는 각 삼각형의 최소한 한 에지가 사분면 근접 그래프 $QNG(S)$에 포함됨을 밝히고, $QNG(S)$의 각 에지에 대응되는 $DT(S)$의 각 삼각형을 레인지 트리 탐색으로 찾아내는 방법을 논의한다. 이 방법은 $DT(S)$의 각 삼각형을 $QNG(S)$의 각 에지로부터 개별적으로 구축하기 때문에 병렬처리가 용이하게 된다. 또한, 이 과정에서 직선간의 교차점 계산 대신 거리비교를 하기 때문에 수치오차가 적고 안정적이다.

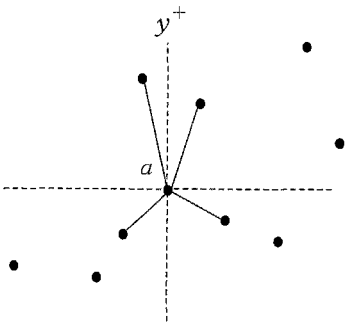

그림 3 점 a의 사분면 근접 점

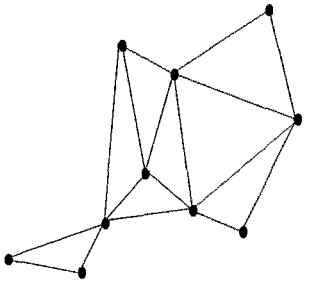

그림 4 사분면 근접 그래프 $QNG(S)$

본 논문은 다음과 같이 구성된다. 2절에서는 $DT(S)$와 $QNG(S)$의 기하학적 특성을 살펴본다. 3절에서는 알고리즘을 논의하고 성능분석을 한다. 마지막으로, 4절에서는 결론을 내린다.

2. QNG와 DT의 기하학적 특성

두 점 a, b 사이의 L_∞ 거리는 $d(a,b)=\max(|a_x-b_x|, |a_y-b_y|)$로 정의되며 단위거리형 (한 점에서 단위거리에 있는 점들의 집합)은 밑변이 x 축에 평행한

정사각형이 된다. 두 점 a, b 사이의 L_1 거리는 $d(a,b)=|a_x-b_x|+|a_y-b_y|$로 정의되며 단위거리형은 L_∞ 단위거리형을 45° 회전 한 모양이 된다. 앞으로 별도의 언급이 없으면 거리는 L_∞로 간주하며 논의한 결과는 L_1 거리 상에서도 바로 적용된다.

세 점 a, b, c가 변에 위치하는 정사각형을 $\square(a,b,c)$로 나타내기로 하자. $\square(a,b,c)$가 S의 점을 내부에 포함하지 않으면 비어있다고 하고 $\square_E(a,b,c,S)$로 나타내기로 한다. 임의의 세 점 $a, b, c \in S$에 대하여 한 삼각형 (a,b,c)가 $DT(S)$에 포함되기 위한 필요충분 조건은 $\square_E(a,b,c,S)$가 존재하는 것이다 [8] (그림 5 참조). 따라서, $DT(S)$는 세 점 이상이 변에 위치한 비어있는 정사각형들을 찾음으로서 바로 구축 될 수 있다. 여기서, 두 점 이상이 정사각형의 한 변에 있으면 $\square_E(a,b,c,S)$가 유일하지 않을 수 있으며, 따라서 $DT(S)$도 유일하지 않을 수 있다. 이 절에서는 논의를 간결하게 하기 위하여 두 개 이상의 점이 정사각형의 한 변 있을 수 없는 것으로 가정한다. 이 조건을 만족하지 않는 경우 (degenerate case)에 대한 처리 방법은 3절에서 논의된다.

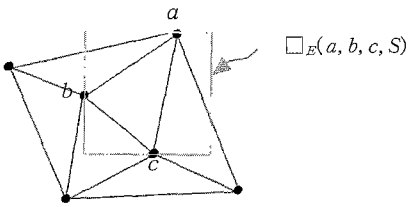

그림 5 $\square_E(a,b,c,S)$ if and only if $(a,b,c) \in DT(S)$

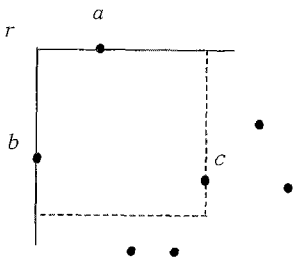

그림 6 $\square_E(a,b,c,S)$, $(b,a) \in QNG(S)$, $d(r,c) \geq \max(d(r,a), d(r,b))$.

$DT(S)$의 한 삼각형 (a,b,c)에 대응되는 $\square_E(a,b,c,S)$을 생각하여 보자. 그림 6에서와 같이 두 점 a, b가 $\square_E(a,b,c,S)$의 서로 인접한 변에 놓여있고 $d(r,a) \leq d(r,b)$ 라고 가정하자. 그러면, $\square(r,b,a)$는 $\square_E(a,b,c,S)$에 포함되므로 비어있게 된다. 또한, $d(r,a) \leq d(r,b)$ 이기 때문에 b는 $\square_E(r,b,a,S)$의 한 꼭지점에 위치하여 a는 b의 북동 사분면 근접 점이 된다.

정리 1: $DT(S)$에 속한 한 삼각형 (a,b,c)의 두 점 a, b가 $\square_E(a,b,c,S)$의 서로 인접한 변에 놓여있으면 $(a,b), (b,a)$ 중의 적어도 하나는 $QNG(S)$에 속하게 된다.

증명: 위의 논의에 따름.

세 점이 한 정사각형의 각각 다른 변에 놓여 있으면 적어도 한 쌍의 두 점은 정사각형의 인접한 변에 위치하게 된다. 따라서, 각 삼각형 $(a,b,c) \in DT(S)$의 적어도 한 에지가 $\square_E(a,b,c,S)$에서 인접하게 되며, 정리 1에 의하여 $QNG(S)$에 속하게된다. 그러므로, $QNG(S)$의 각 에지 (a,b)에 대응되는 $\square(a,b,_,S)$를 찾으면 바로 $DT(S)$를 구축하게 된다. 임의의 두 점 $a, b \in S$에 대하여, 두 점 a, b가 인접하여 위치한 $\square_E(a,b,_,S)$를 찾는 방법은 다음과 같다.

두 점 a, b가 $\square(a,b,_)$의 인접한 두 변에 있으면 $\square(a,b,_)$의 한 꼭지점 r은 a, b를 통과하는 수직선과 수평선의 교점 (a_x, b_y) 또는 수평선과 수직선의 교점 (b_x, a_y)이 된다. 두 점 a, b에 의하여 결정되는 $\square(a,b,_)$의 꼭지점을 a, b의 수직교점이라 하자. 두 점 $a, b \in S$의 한 수직교점 r에 대한 사분면 $\llcorner a,r,b$의 근접 점이 $c \in S$ 이고 $d(r,c) \geq \max(d(r,a), d(r,b))$ 이면 $\square(a,b,c)$가 비어있게 되어서 (a,b,c)는 $DT(S)$에 속하게 된다. 그림 6의 예를 보면, 두 점 a, b의 수직교점 r의 남동 근접 점은 c 이고 $d(r,c) \geq \max(d(r,a), d(r,b))d(r,b))$이 성립하므로 $\square(a,b,c)$는 비어있게 된다. 여기서, $d(r,c) < \max(d(r,a), d(r,b))$이면 r을 한 꼭지점으로 하여 세 점 a, b, c를 변에 포함하는 정사각형이 존재 할 수 없다. 또한, $\square(a,b,c)$가 비어있지 않으면 c가 a의 남동 근접 점이 될 수 없다.

정리 2: 두 점 $a, b \in S$의 한 수직교점 r의 사분면

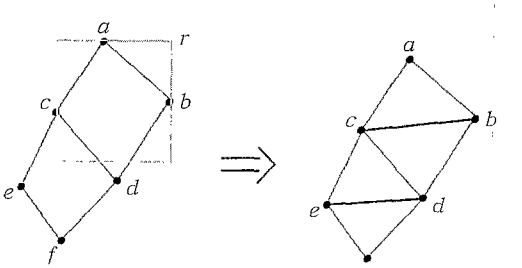

그림 7 $QNG(S)$로부터 $DT(S)$ 구축의 예

$\triangle a, r, b$에 대한 근접 점이 $c \in S$ 일 때 $d(r, c) \geq \max(d(r, a), d(r, b))$ 는 $(a, b, c) \in DT(S)$의 필요충분 조건이다.

증명: 위의 논의에 따름.

3. 알고리즘 및 분석

2절의 논의 결과에 따라, 먼저 $QNG(S)$를 구한 다음 $QNG(S)$의 각 에지 (a, b)의 수직교점 r의 사분면 근접 점 c를 구하고 정리 2의 조건 $d(r, c) \geq \max(d(r, a), d(r, b))$을 만족하면 (a, b, c)를 DT에 추가함으로써 $DT(S)$를 구축 할 수 있게된다. 그림 7의 예를 보면, 삼각형 (a, b, c)는 에지 $(a, b) \in QNG(S)$의 수직교점 r의 남서 사분면 근접 점 c가 정리 2의 조건 $d(r, c) \geq \max(d(r, a), d(r, b))$을 만족하므로 DT에 포함된다. 이와 유사하게, 삼각형 (b, c, d), (c, d, e), (e, d, f)도 DT에 포함된다. $QNG(S)$로부터 $DT(S)$를 구축하는 알고리즘은 다음과 같이 구성된다.

Delaunay(S)

1) 사분면 그래프 $QNG(S)$를 구축한다.

2) $DT = \{\ \}$

$QNG(S)$에 속한 각 에지 (a, b)의 두 수직교차점 $r = (a_x, b_y)$, (b_x, a_y)에 대하여

r의 사분면 $\triangle a, r, b$ 근접 점 c가 있고 $d(r, c) \geq \max(d(r, a), d(r, b))$ 이면

$$DT = DT + (a, b, c)$$

r의 사분면 $\triangle a, r, b$ 근접 점이 없으면

$$DT = DT + (a, b)$$

위의 알고리즘 Delaunay에서 1)의 사분면 근접 그래프 구축은 2)의 사분면 근접 점 질의에 필요한 레인지 트리를 구축함과 동시에 구축된다. 또한, 한 사분면에 대한 근접 점 처리 방법은 좌표축의 회전에 의하여 다른 사분면 근접 점에도 적용된다. 다음은 남서 사분면 근접 점에 대한 레인지 트리를 구축하는 방법과 근접 점 질의를 처리하는 방법을 살펴본다. L_∞ 거리의 단위 거리형은 정사각형이므로 한 점을 중심으로 한 사분면 영역 내의 단위거리형의 경계 면은 하나의 수직선과 하나의 수평선으로 구성된다 (그림 8 참조). 따라서, 한 사분면을 단위거리형의 한 꼭지점을 기준으로 두 개의 영역으로 나누면 각 영역내의 단위거리형의 경계 면은 하나의 직선이 된다. 영역내의 단위거리형의 경계 면이 하나의 직선이 되면 그림 3에서와 같이 r_1의 남서서 팔분면 $\cap S =$ (r_2의 남서서 팔분면 $\cup$ r_3의 남서서 팔분면) $\cap S$ 이면 (즉, 영역 (r_1, r_2, t, r_3)이 비어있으면) r_1의 남서서 팔분면 근접 점은 r_2의 남서서 팔분면 근접 점 d 또는 r_3의 남서서 팔분면 근접 점 f가 되므로 일반적인 레인지 트리 방법을 남서서 팔분면 근접 점 질의에 적용 할 수 있다. 마찬가지로 남남서 팔분면에 대해서도 일반적인 레인지 트리 방법을 적용 할 수 있다. 남서 사분면 근접 점은 남서서 팔분면 근접 점 또는 남남서 팔분면 근접 점이 되므로 두 개의 팔분면 근접 점에 대한 레인지 트리로 남서 사분면 근접 점 질의를 처리 할 수 있다.

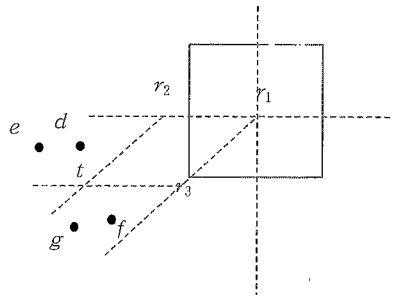

그림 8 점 r_1를 중심으로 한 L_∞ 단위거리형 및 영역분할: L_∞거리에서, $d(r_1, d) = d(r_1, r_2) + d(r_2, d)$ 이고 $d(r_1, f) = d(r_1, r_3) + d(r_3, f)$가 된다.

레인지 트리는 $O(n)$ 프로세서로 $O(\log n)$ 시간에 구축 할 수 있으며 이 트리 상에서 각 질의는 $O(1)$ 프로세서로 $O(\log n)$ 시간에 처리 할 수 있다 [1,15]. 단계 1)의 사분면 그래프 $QNG(S)$는 S에 있는 n개의 각 점에 대하여 사분면 근접 점 질의를 함으로써 구축 될 수 있으며 $O(n)$ 프로세서로 $O(\log n)$ 시간에 처리된다. $QNG(S)$의 각 에지 당 2 개의 수직교차점이 있으며 $QNG(S)$의 에지 수는 $O(n)$ 이기 때문에 단계 2)에서 전체 근접 점 질의는 $O(n)$ 프로세서로 $O(\log n)$ 시간에 처리된다. 따라서 위의 알고리즘은 $O(n)$ 프로세서로 $O(\log n)$ 시간에 처리된다.

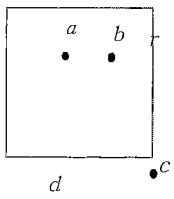

그림 9 c가 옆 변에 있는 경우

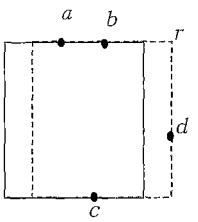

그림 10 c가 반대편 변에 있는 경우

2 절에서 언급하였던 degenerate case (2 점 이상이 정사각형의 한 변에 놓여있는 경우)처리 방법에 대하여 논의하여 보자. $DT(S)$에 있는 한 삼각형 (a, b, c)의 두 점 a, b가 정사각형 $\square_E(a, b, c, S)$의 위쪽 변에 동시에 놓여 있는 경우를 생각하여 보자. 먼저, 점 c가 인접한 변에 놓여있는 경우를 생각하여 보자 (그림 9 참조). 점 a가 점 b의 좌측에 위치하며, 점 c는 우측 변에 놓여 있다고 가정하자. 두 점 b, c가 인접한 변에 위치하므로 정리 1에 의하여 (b, c) 또는 (c, b)가 $QNG(S)$에 속하게 된다. 정리 2에 따라, b, c의 한 수직 교차점 r의 남서 근접 점이 d 일 때 $d(r, d) \geq \max(d(r, a), d(r, c))$ 이면 네 점 a, b, c, d가 비어있는 한 정사각형의 변에 위치하게 되어 네 점 a, b, c, d의 삼각분할이 $DT(S)$에 속하게 된다. 여기서, r의 남서 근접 점이 존재하지 않으면 삼각형 (a, b, c)가 $DT(S)$에 속하게 된다. 점 c가 a, b가 위치한 변의 반대편 변에 놓여있는 경우를 생각하여 보자 (그림 10 참조). $\square_E(a, b, c, S)$를 우측으로 이동하여 좌측 상단 꼭지점이 점 a를 만나기 전에 최초로 우측 변과 만나는 점을 d라 하자. d가 존재하면, 앞의 경우와 유사하게 (b, d) 또는 (d, b)가 $QNG(S)$에 속하게 되고 $d(r, c) \geq \max(d(r, b), d(r, d))$ 이면 네 점 a, b, c, d의 삼각분할이 $DT(S)$에 속하게 된다. 또한, d가 존재하지 않으면 삼각형 (a, b, c)가 $DT(S)$에 속하게 된다.

위의 논의에 따라, 알고리즘 Delaunay를 일부 변형하여 2 점이 한 변에 위치하는 경우를 처리 할 수 있으며, 유사한 방법으로 3 점 이상이 한 변에 놓여있는 경우도 처리 할 수 있다. 단지, 2 점 이상이 한 변에 놓여 있는 경우 $DT(S)$는 유일하지 않을 수 있다.

4. 결 론

본 논문에서는 영역별 근접 그래프와 레인지 트리를 이용하여 평면 위의 n 개의 점에 대한 $L_\infty(L_1)$ 거리 상의 디루니 삼각분할 (따라서, 보로노이 다이아그램)을 $O(n \log n)$의 순차계산 시간과 CREW-PRAM 상에서 $O(n)$의 프로세서로 $O(\log n)$의 병렬처리 시간에 처리하는 방법을 고안하였다. 디루니 삼각분할은 최소한 $\Omega(n \log n)$의 계산시간을 요구하기 때문에 이 방법의 처리시간은 최적이 된다. 또한, 이 방법은 직선간의 교차점 계산 대신 거리비교를 하기 때문에 수치오차가 적고 안정적이며 구현이 용이하여 실용적이다.

본 논문과 관련된 미해결 문제들을 살펴보면, L_2 거리 상의 최적 병렬처리 디루니 삼각분할 방법은 아직도 중요한 미해결 문제로 남아 있으며, L_2 거리 상의 영역별 근접 그래프 문제는 병렬처리 뿐만 아니라 순차계산도 최적시간 알고리즘이 알려져 있지 않다 [3,6].

참 고 문 헌

[1] M.J. Atallah, R. Cole, and M.T. Goodrich, Divide-and-Conquer: A technique for Designing Parallel Algorithms, *28th* IEEE *Symp. on Foundations of Computer Science*, Tech. Rep. 665, Dept. of Comp. Sci., Purdue Univ., 151-160, 1987.

[2] A. Aggarwal, B. Chazelle, L. Guibas, C. Ó' Dúnlaing and C. Yap, Parallel Comutational Geometry, *Algorithmica*, 293-327, 1988.

[3] R. Cole, M.T. Goodrich, Optimal Parallel Algorithms for Polygon and Point-Set Problems, *Proc. of the Fourth Ann. Symp. on Computational Geometry*, 201-210, 1988.

[4] K. Clarkson, Approximation Problems for Shortest Path Motion Planning, *Proc. 19th Ann. ACM Symp. Theory of Computing*, 56-65, 1987.

[5] H.N. Gabow, J.L. Bentley and R. E. Tarjan, Scaling and Related Techniques for Geometry Problems, *Proc. 16st Ann Symp. Theory of Computing*, 135-143, 1984.

[6] L.J. Guibas and J. Stolfi, On Computing all North-east Nearest Neighbors in the L_1 Metric, *Info. Process. Lett. 17*, 219-223, 1983.

[7] J. Katajainen, The Region Approach for Computing Relative Neighborhood Graphs in the L_P Metric, *Computing* **40**, 147-161, 1988.

[8] D.T. Lee, Two-Dimensional Voronoi diagrams in L_P-Metric, *J.ACM* 27(4), 604-618, 1980.

[9] D.T. Lee and C.K. Wong, Voronoi Diagrams in $L_1(L_\infty)$ Metrics with 2-Dimensional Storage Applications, *SIAM J. Comput.* 9(1), 200-211, 1980.

[10] E. Papadopoulou and D.T. Lee, Critical Area Computation via Voronoi Diagrams, *IEEE Transactions on Computer Aided Design of*

Integrated Circuits and Systems, 18(4), 463-474, 1999.

[11] W. Preilowski and W. Mumbeck, A Time-Optimal parallel Algorithm for the Computing of Voronoi diagrams, Lecture Notes *in Computer Science, Springer Verlag*, 424-433, 1988.

[12] P. Su, R.L. Drysdale, A Comparison of Sequential Delaunay Triangulation Algorithms, *Computational Geometry* 7, 361-385, 1997.

[13] K.J. Supowit, The Relative Neighborhood Graph, with an Application to Minimum Spanning Trees, *J.ACM* **30**(3), 428-448, 1983.

[14] Y.C. Wee and S. Chaiken and S. S. Ravi, Rectilinear Steiner Tree Heuristics and Minimum Spanning Tree Algorithms Using Geographic Nearest Neighbors, *Algorithmica* 12, 421-435, 1994.

[15] D.E. Willard and Y. C. Wee, Quasi-valid Range Querying and its Implications for Nearest Neighbor Problems, *Proceedings of the Fourth Annual Symp. on Computational Geometry,* 34-43, 1988.

[16] A.C. Yao, On Constructing Minimum Spanning Trees in k-dimensional Spaces and Related Problems, *SIAM J.* Comput. **11**(4), 721-736, 1982.

위 영 철
정보과학회논문지 : 시스템 및 이론
제 28 권 제 2 호 참조