도로망이 설치된 평면에서의 보로노이 다이어그램

배 상 원*, 좌 경 룡

한국과학기술원 전자전산학과 전산학전공

{swbae,kychwa}@jupiter.kaist.ac.kr

Voronoi Diagrams with a Transportation Network

Sang Won Bae*, Kyung-Yong Chwa

Division of Computer Science, Department of Electrical Engineering and Computer Science

Korea Advanced Institute of Science and Technology

요 약

본 논문에서는 유클리드 평면상에 도로망이 주어져 있어서 여행자들이 그 도로들을 이용하여 더욱 빠르게 이동할 수 있을 경우를 가정한다. 이 때, 두 점 사이의 거리는 기하학적 직선거리가 아닌 주어진 도로들을 이용하여 두 점 사이를 이동할 때 필요한 최소시간으로 측정한다. 본 논문에서는 이러한 새로운 거리 척도를 고려할 때에 보로노이 다이어그램이 어떤 특성을 갖는가를 연구하며 그것을 이용하여 보로노이 다이어그램을 효율적으로 계산하는 알고리즘을 제시한다. 이 알고리즘은 $O(nm^2 \log n + m^3 \log m)$의 시간과 $O(m(n+m))$의 공간을 필요로 한다. 이 때, n은 주어진 싸이트의 개수이고 m은 주어진 도로의 개수이다.

1 서론

유클리드 평면상에서의 보로노이 다이어그램은 매우 고전적인 문제로써 수많은 사람들에 의해 연구되어 왔고 그만큼 많은 수의 변형과 그 응용 또한 이미 밝혀져 있다. [4] 본 논문에서는 유클리드 평면상에 도로망이 주어져 있어서 여행자들이 그 도로들을 이용하여 더욱 빠르게 이동할 수 있을 경우를 가정한다. 기본적으로 여행자들은 도로의 어느 점에서든지 출입이 가능하다. 도로망은 직선의 간선을 갖는 평면 그래프 $G = (V, E)$로 주어진다. 여기서 E는 도로의 집합으로 볼 수 있으며 V는 도로들의 끝점들의 집합이다. G는 평면상에 임베드되어 있어서 각 도로는 평면상의 선분으로 표현할 수 있다. 또한 각 도로 e에는 개별적인 속도 $v(e)$가 주어지며, 도로 밖에서의 속도는 1이라고 가정한다. 보통의 유클리드 평면상에서의 거리는 두 점사이의 직선거리(즉, L_2 norm)를 이용하지만 본 논문에서는 직선거리가 아닌, 도로를 사용하면서 갈 수 있는 가장 빠른 시간을 "거리"로 간주한다. 이것을 도로망 G가 주어진 경우의 거리로 정의하며 d_G로 표시한다. 이런 가정하에 평면상의 n개의 점들의 집합 S가 주어질 경우 S에 대한 보로노이 다이어그램 $V_G(S)$를 정의할 수 있으며, 본 논문에서는 $V_G(S)$를 $O(nm^2 \log n + m^3 \log m)$시간 안에 계산할 수 있는 알고리즘을 제안한다. 여기서 m은 주어진 도로망의 도로의 개수이다.

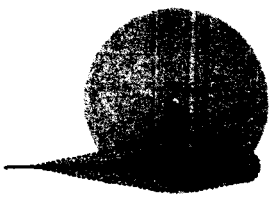
관련된 연구가 이전에도 있었다. Hurtado 등 [5]은 이런 도로망이 있는 경우의 보로노이 다이어그램 문제를 처음 제안했으며 유클리드 평면상에 무한히 긴 도로가 한 개 놓여져 있을 경우에 대해 문제를 해결하였다. Abellanas 등 [1]과 Aichholzer 등 [3]은 L_1 평면상에 도로망이 있을 경우에 대해 연구하였는데 특히 후자의 논문에서는 모든 도로가 x축 또는 y축과 평행하고 모든 도로의 속도가 동일한 경우에 대하여 문제를 해결하였으며 이것이 이전의 가장 일반화된 문제에 대한 해결이었다. 본 논문에서는 이전의 결과들 보다 훨씬 더 일반적인 경우인, 유클리드 평면상에 도로망이 직선의 간선을 갖는 평면 그래프로 주어지고 각 도로의 속도도 개별적으로 주어지는 경우를 가정한다.

2 기본 아이디어

도로망이 있는 경우 일반적인 방법으로는 보로노이 다이어그램을 계산하기가 쉽지 않다. 널리 알려진 도구로 *abstract Voronoi diagram* [6] 이 있다. 하지만 이것은 두 점 사이의

bisector가 어떤 조건을 만족시켜야 하는데 도로망이 있는 경우에는 필요한 조건을 만족시키지 못하는 경우를 통해 쉽게 찾을 수 있다.

그렇기 때문에 우리는 더 세밀한 관찰을 통해 다른 접근을 시도해야 한다. 우리는 이 과정을 수월하게 하기 위해 wavefront model을 도입한다. Wavefront model이란 보로노이 다이어그램을 해석하는 또 다른 방법이다. Wavefront model에서는 각 점 싸이트에서 시간 0을 기점으로 어떤 전파를 발산한다고 생각한다. (유클리드 평면에서는 이것이 원의 형태로 뻗어나갈 것이다.) 각 점으로부터 뻗어나가는 전파는 서로 충돌할 것이고 이때에 뻗기를 멈춘다. 이렇게 전파가 멈추는 위치가 각 보로노이 셀의 경계를 만들게 되는 것이다. 물론, 이것은 유클리드 평면상에서도 매우 잘 동작한다. 이것을 도로가 있는 경우에 적용하면 그림 1과 같은 모양을 얻을 수 있다. 점에서 뻗어 나오는 파는 도로에 닿기 전에는 원형을 유지한다. 그러다가 도로에 닿으면 양쪽으로 뾰족한 모양의 파가 도로가 지원하는 속도만큼의 빠르기로 뻗어나간다. 이 모양에 대한 더 자세한 설명이 필요하다면 이전 논문 [2,5]을 참조하기를 바란다.

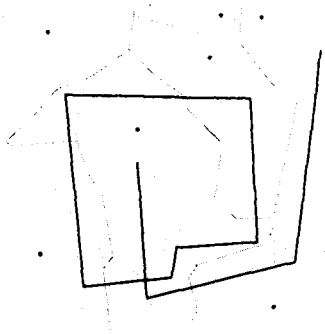


그 림 1: 도로가 있는 경우 점으로부터 생성된 파형

그림 1의 전파 모양을 살펴보면 그것을 세 부분으로 나누어진다는 것을 알 수 있다. 하나는 큰 원의 파형이고 나머지 둘은 도로에서 양쪽으로 뻗어나가는 바늘 같은 모양을 하고 있는 파형이다. 본 논문에서 가장 핵심적인 아이디어는 이 바늘 모양의 파형을 어떤 독립적인 새로운 싸이트로 간주한다는 것이다. 이 새롭게 만들어진 싸이트로부터 새로운 파형이 뻗어나오고 이것은 또 다른 도로에 부딪혀서 또 다른 바늘 모양의 파형을 만들어 낼 것이다.

이전의 일반화된 싸이트는 이런 바늘 모양의 파형을 만드는 것은 없었다. 이 바늘 모양의 파형은 어떤 시작점이 있고 한 방향으로 빠르게 전진하며 그것이 끝나는 점(여기서는 도로의 끝점)이 존재한다. 우리는 이런 더욱 일반화된 싸이트를 "바늘(needle)"이라고 정의한다. 바늘은 이미 알려진 다른 일반화된 싸이트들; 예를 들어 점, 선분에 가산 가중치가 주어진 싸이트들을 표현할 수 있다. 위와 같은 설명을 통해 도로망이 놓여있는 경우 점 싸이트의 보로노이 셀은 여러 개의 바늘들의 셀의 합집합으로 나타낼 수 있음을 알 수 있다.

이런 사실을 이용하기 위해서 abstract Voronoi diagram을



그 림 2: $V_G(S)$. 굵은 검은 선분은 도로, 점은 싸이트, 검은 곡선은 $V_G(S)$의 간선을 나타낸다. 점선은 바늘들의 보로노이 다이어그램의 간선이다.

이용하여 바늘들의 보로노이 다이어그램을 계산할 수 있는 방법을 마련한다. 두 개의 바늘의 보로노이 다이어그램은 각각의 보로노이 영역이 연결되지 않는 경우가 발생할 수 있는데 그런 경우를 두 바늘이 *piercing*하다라고 하고 그렇지 않은 경우는 *non-piercing*하다라고 부르기로 한다. 바늘들의 집합 S안의 모든 바늘들이 서로 non-piercing하다면 그 보로노이 다이어그램은 abstract Voronoi diagram임을 증명할 수 있다. Abstract Voronoi diagram을 최적의 시간과 공간 안에서 계산하는 방법은 여러 가지가 이미 소개되어 있다. [6,7]

정 리 1 *n개의 바늘들의 집합 S안의 모든 바늘들이 서로 non-piercing하다면 그 보로노이 다이어그램은 $O(n \log n)$시간 안에 $O(n)$의 공간을 이용하여 계산할 수 있다.*

3 알고리즘

이번 장에서는 우리의 알고리즘을 설명한다. 알고리즘은 크게 세 단계로 나누어 볼 수 있다. 1단계에서는 주어진 도로망과 점 싸이트의 집합 S로부터 필요한 바늘들 S를 추출한다. 2단계에서는 직접 S의 보로노이 다이어그램 $V(S)$를 계산하고, 3단계에서는 $V(S)$의 단순한 영역들을 합치는 과정으로써 $V_G(S)$를 구한다. 2단계와 3단계는 이미 그 과정을 이미 설명하였거나 간단한 과정으로 여기서는 1단계를 설명하는 데에 주력하겠다.

1단계는 2 장에서 설명했던 wavefront model을 시뮬레이션하면서 동작한다. 이것은 파가 뻗기 시작하는 시간 0에서부터 시간 순으로 일어나는 이벤트들을 처리함으로써 가능하다. 우리는 두 가지의 이벤트를 정의한다. 하나는 새로운 바늘이 생성될 때에 일어나고 다른 하나는 이런 바늘들로부터 나오는 파형이 도로의 끝점에 닿을 때 일어난다. 각각을 *생성 이벤트*와 *정점 이벤트*라 부르겠다. 이 이벤트들을 처리하기 위해 두 가지의 데이터 구조가 사용된다. 하나는 이벤트 큐인 Q로 이벤트들을 저장하고 가장 먼저 일어나는 이벤트를 빠

른 시간 안에 추출할 수 있는 일종의 우선순위 큐이다. 다른 하나는 균형잡힌 이진 검색 트리로 각 도로에 하나씩 필요하며 각 트리에는 그 도로위에 만들어진 바늘들을 출발위치 순으로 정렬하여 저장한다. 이 트리는 $\mathcal{T}_1, \mathcal{T}_2, \cdots, \mathcal{T}_m$ 로 표시하며 m개의 도로 $E = \{e_1, \cdots, e_m\}$에 각각 관계한다.

이제 각 이벤트를 어떻게 처리하는지를 설명한다. 먼저 생성 이벤트 b가 도로 e_i위에 발생 하면 b와 관련된 바늘 p의 유효성을 검사하여 유효하다면 $\mathcal{T}_i$에 추가한다. 유효성이란 바늘이 파형을 발산할 수 있는가이다. 만약 이전에 생성된 다른 바늘에서 나온 파형이 p의 생성위치를 이미 쓸고갔다면 p는 유효하지 않다. 이것은 $\mathcal{T}_i$에 저장된 바늘들을 검사해보면 쉽게 알 수 있다. 만약 p가 유효하다면 현재 시간부터 파형을 발산하기 시작할 것이고 이것은 언젠가 다른 도로들의 끝점에 닿을 것이다. 따라서, 이 때에 정점 이벤트들이 언제 어디서 발생할 지를 미리 계산하여 Q에 넣는다.

정점 이벤트가 어떤 도로의 끝점 v에서 발생했다면, 도로의 끝점을 가산 가중치가 주어진 점 싸이트로 간주하여 다른 모든 도로들에 대해 생성 이벤트를 계산하여 Q에 넣는다.

1단계 알고리즘을 정리해보면 1)S안의 모든 싸이트에 대해 모든 도로에 대한 생성 이벤트를 계산하고 모든 도로의 끝점에 대해 정점 이벤트를 계산하여 Q에 넣고, 2)Q에서 가장 빨리 일어나는 이벤트를 추출하여 처리하고 3) Q가 비게 되면 $\mathcal{T}_i$들로부터 바늘들을 추출하여 그것을 S로 한다. 한가지 더 언급해야 할 것은 S를 서로 non-piercing하게 만들어야 한다는 것인데 이것은 $\mathcal{T}_i$에서 서로 이웃하는 바늘들을 고려하여 각 바늘의 끝점을 조정함으로써 쉽게 이루어진다.

4 분석

3 장에서 제시한 알고리즘을 분석하기 위해 몇 가지 보조정리가 필요하다.

보조정리 2 *어떤 도로 e위에 생성된 바늘 p로부터 도로 e'에 생성된 바늘 q가 유효하다면, p의 파형이 도로 e 또는 e'의 끝점을 쓸고 지나간다.*

보조정리 3 $m = |E|, n = |S|$ *일 때,* $|\mathcal{S}| = O(m(n+m))$.

이 두 개의 보조정리를 통해 우리는 매우 자세한 분석을 할 수 있게 된다. 1단계에서 유효한 정점 이벤트가 정확히 $|V|$개라는 점을 이용하면 S안의 최대 이벤트의 개수는 $O(m(n+m))$이 된다. 또한, Q와 $\mathcal{T}_i$의 기본 연산은 모두 $O(\log(m(n+m)))$ 시간이 걸린다. 따라서 1단계는 $O(m^2(n+m)\log(m(n+m))) = O(nm^2 \log n + m^3 \log m)$시간이 필요함을 보일 수 있다. 2단계는 정리 1에 의해 $O(m(n+m)\log(m(n+m)))$만큼의 시간이 필요하다. 3단계는 각 영역을 단순히 합하는 과정이므로 $V(S)$의 크기에 선형함수 만큼의 시간만이 필요하다. 이런 논의에 의해 다음의 결론에 도달할 수 있다.

정 리 4 *G를 m개의 직선도로를 가진 유클리드 평면상의 도로망이라 하고, S를 평면상의 n개의 점이라고 하자. 이 때, 그 보로노이 다이어그램 $V_G(S)$는 $O(nm^2 \log n + m^3 \log m)$ 시간 안에 계산할 수 있으며 $O(m(n+m))$ 만큼의 공간이 필요하다.*

5 결론

우리는 유클리드 평면상에 도로망이 주어질 경우에 점들의 보로노이 다이어그램을 계산하는 알고리즘을 제시하였다. 하지만 우리의 알고리즘은 내부적으로 바늘이라는 매우 일반화된 싸이트를 다루고 있기 때문에 싸이트가 선분이나 가중치가 주어진 점과 같은 경우라도 알고리즘에 큰 수정없이 바로 적용할 수 있다는 큰 장점이 있다.

우리가 다룬 문제의 시간상의 이론적 하한과 알고리즘의 개선 등은 추후과제로 남아 있다.

참 고 문 헌

[1] M. Abellanas, F. Hurtado, C. Icking, R. Klein, E. Langetepe, L. Ma, B. Palop, and V. Sacristán. Proximity problems for time metrics induced by the l_1 metric and isothetic networks. *IX Encuetros en Geometria Computacional*, 2001.

[2] M. Abellanas, F. Hurtado, V. Sacristán, and B. Palop. Voronoi diagram for services neighboring a highway. *Information Processing Letters*, 86, 2003.

[3] O. Aichholzer, F. Aurenhammer, and B. Palop. Quickest paths, straight skeletons, and the city voronoi diagram. In *Proceedings of 18th SoCG*, pages 151–159, New York, 2002. ACM Press.

[4] F. Aurenhammer and R. Klein. Voronoi diagrams. In J.-R. Sack and J. Urrutia, editors, *Handbook of Computational Geometry*. Elsevier, 2000.

[5] F. Hurtado, B. Palop, and V. Sacristán. Diagramas de voronoi con funciones temporales. *VIII Encuetos en Geometria Computacional*, 1999.

[6] R. Klein. *Concrete and Abstract Voronoi Diagrams*. Number 400 in Lecture Notes in Computer Science, LNCS. Springer-Verlag, Berlin, Germany, 1989.

[7] R. Klein, K. Mehlhorn, and S. Meiser. Randomized incremental construction of abstract voronoi diagrams. *Computational Geometry: Theory and Applications*, 3:157–184, 1993.