220
논문번호 07-04-32

Voronoi-Based Search Scheme for Road Network Databases

도로 망 데이터베이스를 위한 보로노이 기반의 탐색 방안

김 대훈(Dae-hoon Kim)[1], 황 인준(Eenjun Hwang)[2]

Abstract

Due to the improved performance and cost of personal mobile devices and rapid progress of wireless communication technology, the number of users who utilize these devices is increasing. This trend requires various types of services be available to users. So far, there have been many solutions provided for the shortest path problem. But, technologies which can offer various recommendation services to user depending on user's current location are focused on Euclidean spaces rather than road network. Thus, in this paper, we extend the previous work to satisfy this requirement on road network database. Our proposed scheme requires pre-computation for the efficient query processing. In the preprocessing step, we first partition the input road network into a fixed number of Voronoi polygons and then pre-compute routing information for each polygon. In the meantime, we select the number of Voronoi polygons in proposition to the scale of road network. Through this selection, the required size of pre-computation is linearly increasing to the size of road network. Using this pre-computated information, we can process queries more quickly. Through experiments, we have shown that our proposed scheme can achieve excellent performance in terms of scheduling time and the number of visited nodes.

요 약

개인용 이동형 단말기의 개선된 성능과 비용, 그리고 무선 통신 기술의 비약적인 발전으로 인하여, 이를 이용하는 사용자들의 수가 빠른 속도로 늘고 있다. 그에 따라 사용자들에게 다양한 서비스를 제공할 수 있는 기술이 요구하고 있는 시점이다. 현재까지의 연구를 통해 사용자가 필요로 하는 최단 경로 찾기 등의 기술은 많은 연구가 이루어져 있다. 하지만 사용자의 현재 위치에 따라 여러 가지 추천 서비스를 제공할 수 있게 하는 기술은 우리가 필요로 하는 도로 망에서가 아닌 Euclidean spaces 에 집중되어 있다. 따라서 우리는 기존의 연구를 확장시켜, 도로 망에서 이러한 요구를 충족시킬 수 있는 방법을 제안한다. 우리가 제안하는 시스템은 질의에 대한 응답을 하기 위해 전처리 단계를 필요로 한다. 이 단계에서는 먼저 전체 도로 망을 몇 개의 Voronoi 다각형으로 나누고, 나누어진 각각의 Voronoi 다각형들에 대한 정보를 계산한다. 이러한 과정에서 도로 망의 규모에 맞춰 자동으로 Voronoi 다각형의 개수를 결정하게 한다. 이를 통해 전체 도로 망의 크기가 변경되더라도, 전처리 단계 정보를 저장하는 공간이 예측 가능하도록 선형적으로 증가되게 하였다. 실제 질의 응답과정에서는 미리 계산된 정보를 이용하여 사용자들에게 빠른 속도의 서비스를 제공 할 수 있게 한다. 실험을 통하여 제안된 시스템이 도로 망에서 최근접 질의와 영역 질의를 효과적으로 처리 하여 탐색 시간과 방문 노드 수에서 많은 이점이 있음을 보인다.

주제어 : 보로노이 다이어그램, 최근접질의, 영역질의, 도로 망 데이터베이스

Keywords : Voronoi diagram, kNN query, Range query, Road network

---

과학재단 특정연구개발사업(유전자원활용사업단)의
식물유전자원정보은행구축 과제(no. BDM0100211)의
지원에 의하여 수행되었습니다.

[1] 고려대학교 전자컴퓨터공학과 석사 과정
[2] 고려대학교 전기전자전파공학과 부교수, 교신저자

I. 개요 및 관련 연구

공간 질의는 우리의 실생활에 매우 유용하고 빈번하게 쓰이는 질의 형태이며, 이에 관련된 연구가 현재까지 매우 활발히 진행되고 있다. 최근에는 개인용 이동형 단말기의 발전과 더불어 GPS(Global Positioning System)를 이용한 위치 관리 기술의 발달에 힘입어, 사용자의 현재 위치에 따라 다양한 정보와 추천 서비스를 제공해 주는 위치 기반 서비스(LBS: Location Based Services)[1] 같은 기술이 주목 받고 있는 상황이다. 하지만, 공간 질의의 대표적인 예인 최근접 질의 등에 대한 연구는 지금까지 Euclidean spaces에 제한되어 왔다.

R-tree[2][3][4]는 Euclidean spaces에서 공간 질의에 대한 응답을 하기 위한 대표적인 인덱싱 구조이다. 이 방법은 B-tree[5]와 비슷하지만 다차원 공간에 대한 인덱싱을 지원한다. 예를 들면 지리학에서 R-트리는 "현재 위치에서 200km 이내의 모든 도시를 찾아라" 와 같은 질의에 대해 빠르게 답을 줄 수 있다. 이 자료 구조는 공간을 최소 경계 사각형(MBR: Minimum Bounding Rectangle)들로 분할하여 저장한다. 이러한 최소경계 사각형들은 서로 겹칠 수도 있으며, 상위 레벨의 최소경계 사각형은 하위 레벨의 최소경계 사각형들을 포함하는 계층적인 트리 구조로 구성되어 있다. R-트리의 각 노드는 미리 정의된 범위 내에서 유동적인 개수의 자식 노드들의 정보 (MBR과 포인터)를 가진다. R-트리의 저장과 삭제 알고리즘은 가까운 데이터들은 가능한 한 같은 단말 노드 (leaf node)에 두고자 한다. 그렇게 함으로써 R-트리는 조밀한 최소경계 사각형을 유지할 수 있고, 검색 성능이 좋아지게 되는데, 이는 검색 알고리즘들 (intersection, containment, nearest neighbor)이 이 최소경계 사각형들을 이용하여, 하위 레벨의 자식 노드를 검색할 것인지를 결정하기 때문이다. 이러한 R-tree를 이용하여 [6]에서는 최근접 질의를 처리하는 방법을 제안했다. 하지만, Euclidean spaces와 도로 망은 그 차이가 명백하기 때문에, 이러한 기술은 우리가 필요로 하는 도로 망에 직접 적용해서 사용하기에는 문제가 있다.

이러한 배경에서 INE[7]는 도로 망 같은 공간에서 효과적으로 최근접 질의를 처리 할 수 있는 방법을 제안 했다. 하지만 이 방법은 노드들의 밀도가 높지 않는 경우 성능이 매우 떨어지는 단점이 있다. 이러한 단점을 해결 하기 위해 [8]에서는 네트워크 보로노이 다이어그램에 기반하여 최근접 질의를 처리 할 수 있는 방법을 제안했다. 이 방법은 INE에 비해 성능 향상이 있지만, 그 구조가 복잡하여 유지 보수 측면에서 어려움이 있다.

본 논문에서는 위의 두 가지 방법의 단점을 보완할 수 있는 새로운 공간 질의 처리 방안을 제안한다. 우리가 제안하는 방법은 우선 주어진 전체 도로 망을 몇 개의 NVP(Networked Voronoi Polygon)로 나누고, 나누어진 각각의 NVP 에 대해 여러 가지 정보를 미리 계산한다. 이 때, 평가 함수를 통해 도로 망의 형태에 따라 가장 효율이 높은 NVD 를 생성하게 된다. 이것을 통해 노드들의 밀도에 상관 없이 거의 일정한 성능을 보장할 수 있게 하였다. 또한 단 하나의 NVD 를 생성하여 이용하는 것으로 유지 보수 측면에서도 좋은 효과를 기대 할 수 있다.

이 논문의 나머지는 다음과 같이 구성된다. 2 장은 본 연구의 기본이 되는 NVD 에 대해 알아본다. 그리고 3 장은 실제 질의 처리에 필요한 여러 가지 정보를 생성해 놓는 전처리 과정을 알아본다. 4 장에서는 최근접 질의와 영역 질의를 위해 두 가지 알고리즘을 제안한다. 5 장에서는 실험에 대한 결과를 보여주고 마지막으로 6 장에서는 이 논문을 결론 짓는다.

II. 네트워크 보로노이 다이어그램

현재까지 알려진 지역 분할 방법의 종류는 다양하다. 우리가 기존에 제안했던 알고리즘은 그 중에서 네트워크 보로노이 다이어그램[9]이라는 지역 분할 방법을 기반으로 한다. 네트워크 보로노이 다이어그램은 보로노이 다이어그램[10]의 특별한 한 종류로써, 전체 공간을 겹치는 영역 없이 다양한 형태의 다각형들로 나눈다. 따라서 이 방법을 사용한다면, 도로 망과 같이 그 분포 형태가 일정하지 않은 공간을 효율적으로 분할 할 수 있다. 이 장에서는 먼저 네트워크 보로노이 다이어그램과 그 기본이 되는 보로노이 다이어그램에 대해 소개한다.

2.1 보로노이 다이어그램

보로노이 다이어그램은 Delaunay triangulation 과 함께 계산 기하학 분야에 많이 사용되는 효율적인 지역분할 방법이다. 기준점들이 주어지고, 각각의 기준점이 하나만 포함되도록 전체 영역을 분할하고자 할 때, 분할된 영역의 임의의 점이 다른 영역의 기준점에서의 거리보다도, 자신이 속한 영역의 기준까지의 거리가 제일 가깝게 분할하는 방식을

보로노이 다이어그램이라고 한다.

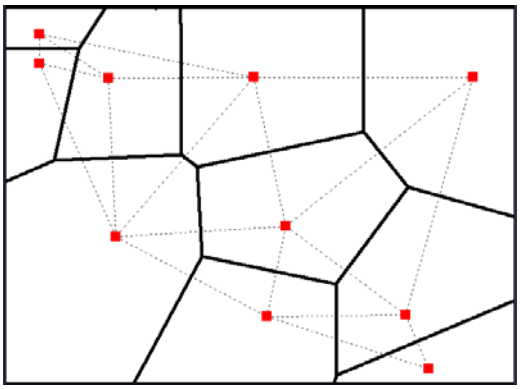


그림 1. 두 가지 공간 분할 방법 예제(Example of two space partitioning methods)

그림 1은 보로노이 다이어그램과 Delaunay triangulation에 따른 공간 분할 방법의 예를 보여준다. 이 그림에서, 점으로 표시된 것이 기준점들이고 실선은 보로노이 다이어그램, 점선은 Delaunay triangulation을 나타낸다. Delaunay triangulation은 보로노이 다이어그램과 듀얼의 관계로, 둘 중 어느 하나만을 찾아낸다면 다른 하나도 쉽게 찾아낼 수가 있다. 보로노이 다이어그램은의 정의는 다음과 같다.

(정의) 집합 $P = \{p_1, p_2, \ , p_n\}$를 한 평면 위에 있는 n개의 기준점들의 집합이라고 하자. 보로누이 다각형 $VP(p_i)$는 다음과 같다.

$$VP(p_i) = \{p \mid dist(p, p_i) \le dist(p,p_j)\} \text{ for } j \neq i, j = 1,...,n$$

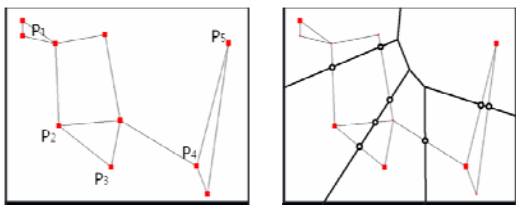
여기서 $dist(p, p_i)$는 기준점 p와 p_i 사이의 Euclidean 거리이다. 그러면 집합 P의 보로누이 다이어그램, VD(P)는 다음과 같이 정의된다.

$$VD(P) = \{VP(p_1),...,VP(p_n)\}$$

이 정의에 따라 각각의 기준점은 자신만의 보로누이 다각형을 가지게 된다. 그리고 각 다각형 내에서 그 다각형 기준점까지의 거리는 다른 다각형의 기준점까지의 거리보다 항상 짧게 된다.

2.2 네트워크 보로노이 다이어그램

네트워크 보로누이 다이어그램은 보로누이 다이어그램의 특별한 경우로, 해당 다이어그램을 생성하는데 모든 노드를 사용하지 않는다. NVD는 interest points라고 불리는 노드 만으로 보로누이 다이어그램을 생성한다. NVD의 예제는 **그림 2**에서 볼 수 있다.



a. 도로 망(Road network) b. NVD

그림 2. NVD 의 예제(An Example of a NVD)

그림 2-(a)는 10 개의 노드를 가진 도로 망의 예를 보여주고 있다. 여기서 우리는 P1 부터 P3 까지 5 개의 노드들을 interest points로 가정한다. **그림 2**-(b)는 5 개의 interest points로 생성된 NVD의 결과를 보여준다. 이 그림에는 원본 로드 네트워크에서는 없었던 진한 줄이 추가된 것을 볼 수 있는데, 이것이 NVD의 edge이다. 이 edge들과 원래 로드 네트워크에 있던 edge들과의 교차점을 경계점(border points)이라 한다. 본 예제에서는 여덟 개의 경계점을 가지게 된다. 그리고 각각의 경계점과 로드 네트워크의 겹쳐진 edge에 연결된 노드들을 crossing points라 한다.

III. 전처리

다양한 질의에 대해 효과적인 응답을 위해, 우리는 앞 장에서 살펴본 방법을 이용해서 전체 로드 네트워크를 몇 개의 작은 보로노이 다각형으로 나누고, 나누어진 각각의 다각형에 대해 전처리 과정을 진행한다. 이 전처리 정보는 실제 라우팅 단계에서 사용되어 보다 빠른 최단 경로 탐색을 가능하게 한다. 이 장에서는 본 논문의 바탕이 되는 보로노이 기반 라우팅 알고리즘의 전처리 단계에 대해 소개한다. 그리고 전처리 단계에서 생성된 정보가 차지하는

공간을 예측 가능하도록 만드는 기법도 제안한다.

전처리 단계 개요

전처리 단계는 실제 라우팅 과정에서 필요한 여러 가지 정보를 미리 계산하여 놓는 과정이다. 이 과정의 상세 내용은 다음의 그림과 같다.

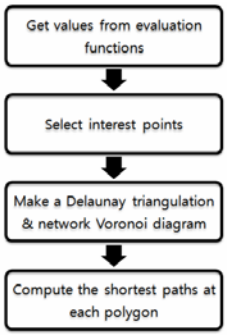
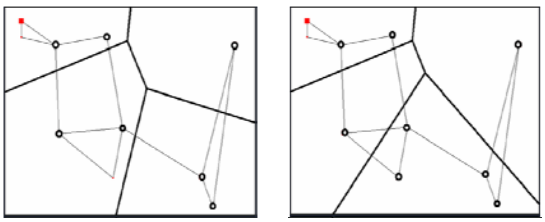


그림 3. 전처리 과정 순서도(The flow of pre-computation)

기존의 연구[8]에서는 특정한 타입의 노드에 대해 각각의 NVD를 생성하는 방법을 사용했다. 하지만 이러한 방법은 노드 타입의 종류가 많아진다면, 그 종류에 대해 각각의 NVD를 전부 생성해야 되기 때문에 많은 비용이 들게 된다. 또한 이런 경우 유지보수를 할 때에도 문제가 될 수 있다. 로드 네트워크가 조금 변경되더라도 그 노드 타입의 NVD를 재생성해야 되기 때문이다. 따라서 우리는 이러한 문제를 해결하기 위해 모든 노드에 대해 하나의 NVD를 생성한다.



a. Case 1 b. Case 2

그림 4. NVD 생성 예제(Examples of making NVDs)

이를 위해, 가장 처음 단계로는 NVD 생성에 필요한 interest points를 선택하게 된다. 특정한 NVD가 생성되었다면 앞서 2.2절에서 살펴본 바와 같이 crossing points도 결정되게 된다. **그림 4**는 interest points의 선택에 따라, 서로 다른 두 가지 모양으로 생성된 NVD의 예제를 보여 준다. 이 그림에서 굵은 점들은 로드 네트워크의 interest points이고 얇은 실선은 로드 네트워크의 edge이다. 그리고 굵은 실선으로 표시된 것은 보로누이 다각형의 edge이다. **그림 4**-(a)는 로드 네트워크의 edge들 중 5 개가 보로누이 다각형의 edge들과 겹치는 것을 볼 수 있고, **그림 4**-(b)는 7 개가 겹치고 있다. 여기서 흰색 점들은 앞서 언급한 바와 같이 로드 네트워크의 노드임과 동시에 crossing points이다. 예제에 소개된 두 개의 그림에서 crossing points의 개수는 각각 7 과 8 이다. 본 예제는 매우 간소화 되어 있는 예제이기 때문에 crossing points의 개수의 차이가 크지 않지만, 실제 로드 네트워크에서는 이 차이가 크게 나타난다. 해당 예제를 통해 interest points의 선택에 따라 다른 모양의 NVD가 생성되고, 결과적으로 선택되는 crossing points의 개수가 달라질 것이라는 것을 알 수 있다.

본 논문에서 제안하는 알고리즘은 실제 경로 탐색 과정에서 이 crossing points 만을 방문해가며 최단 경로를 찾는다. 따라서 crossing points 의 숫자를 최소화 할 수 있는 interest points 의 선택은 매우 중요한 문제이다. 이 것을 위해 우리는 평가함수 w(n)을 사용했다.

$$w(n) = \sum_{i=1}^{N} \frac{1}{dist(n,i)}, \quad for\ i \neq n$$

여기서, n 은 로드 네트워크의 노드이고 N 은 로드 네트워크의 전체 노드 수의 합이다. 또한 dist(x, y)는 노드 x 와 노드 y 사이의 Euclidean 거리이다. 우리는 로드 네트워크의 모든 노드에 대해 이 평가함수로 평가하여 내림차순으로 정렬한다. 그리고 그 순서에 따라 차례로 interest node 로 선택한다. 이를 사용하면 복잡한 로드 네트워크의 사이를 보로누이 다각형의 edge 가 가로지르는 일이 확률적으로 줄어들게 되어 결과적으로는 crossing points 가 적어지게 된다.

그 후에는 선택된 interest points들을 이용해서 Delaunay triangulation을 생성하고[11], 또 이것을 바탕으로 네트워크 보로누이 다이어그램을 생성한다[12]. 마지막으로, 생성된 각각의 보로누이 다각형의 내부에서 여러 가지 정보를 계산한다. 보로누이 다각형 내부의 어떤 노드가 crossing points인지 찾고, 이들간의 최단 경로를 계산하여 저장한다.

표 1. 미리 계산된 다각형 정보(Pre-computed polygon information)

필드	설명
Pid	다각형의 ID
IP	다각형의 Interest point 정보
Edges	다각형의 외곽선 정보
RPoints	Crossing points 를 제외한 inner points
CPoints	Crossing edges 의 Inner points
Paths	Inner crossing points 간의 최단 경로 정보
Tpyes[n]	n Type 노드들의 리스트

전처리 과정

앞 절에서 소개된 전처리 단계를 거치면 다양한 정보가 생성되고 이를 저장하여, 실제 질의 응답을 할 때 이용하게 된다. **표 1**은 이 정보들이 들어갈 구조체를 보여준다.

이 중에서 가장 큰 부분을 차지하는 것이 crossing points들 사이의 최단 경로 정보인 Paths이다. 한 보로누이 다각형에 포함된 노드 수가 n에서 2n으로 증가했을 때를 생각해보자. Paths를 제외한 다른 정보들은 단순히 목록을 가지고 있기 때문에 필요한 용량도 n에서 2n으로 선형적인 증가에 그치게 된다. 하지만 Paths는 그 정보의 양이 기하 급수적으로 늘게 된다. 우선 모든 inner crossing points 쌍간의 최단 경로를 계산하여야 하므로, n^2개의 경로를 저장하면 되었으나 $(2n)^2=4n^2$만큼의 정보를 저장해야 된다. 여기에다가 한 보로누이 다각형에 포함된 노드 수가 많아지면 inner crossing point쌍끼리 최단 경로 길이의 평균값이 길어지게 된다. 따라서 한 polygon에 포함된 노드 수가 n에서 2n으로 증가한다면, 필요한 용량은 n^2에서 $4n^2$이상으로 커지게 된다. 따라서 우리는 이런 식의 기하 급수적인 정보양의 증가를 방지해야 한다. 하지만 반대로, 미리 계산해서 저장해 놓은 정보가 많아진다면, 그만큼 실제 질의에 대한 응답의 속도는 빨라질 것이다. 따라서 전처리 과정 정보를 저장할 공간의 크기를 적절히 유지하면서, 동시에 성능도 일정치 이상을 보장 하기 위해 평가함수 $r(N_i)$를 사용했다.

$$r(N_i) = \frac{Size_{Pre\text{-}Computed} \times Number_{Crossing}}{Number_{Non\text{-}Crossing}}$$

여기서, N_i는 interest points의 개수이다. 어떠한 NVD가 N_i 개의 interest points로 부터 생성되었다고 하면, 그 NVD에 몇 개의 crossing points가 있는지도 정해진다. 앞서 언급한 바와 같이 crossing points의 개수가 작으면, 그만큼 탐색 공간이 좁아졌다는 의미이다. 따라서 우리가 가장 작은 수의 crossing points를 얻는 다면 그만큼 실제 탐색 시간이 줄어들 것이다. 하지만, crossing points의 개수를 적게 하려면 NVP의 개수를 줄여야 하고, 이는 필연적으로 NVP당 노드 수의 증가를 가져오게 되기 때문에 정보의 양이 커지게 된다. 그러므로 우리가 $r(N_i)$이 최소값이 되는 N_i를 찾는 다면, 용량 대비 crossing points의 개수와 non-crossing points의 비율이 가장 좋은 NVD를 찾을 수 있다. 또한 우리는 이 평가함수를 통해 확장성도 보장할 수 있다. 로드 네트워크의 노드 수가 많아지면 평가함수 $r(N_i)$의 값도 변경이 되어 최적화된 N_i를 자동으로 찾을 수 있다.

한편, 지난 연구 [8]에서는 각 노드 종류별로 kNN질의를 처리하기 위해 각각의 노드 종류에 대해 NVD를 생성했다. 이 방법은 노드 종류의 개수가 증가한다면 그만큼 NVD를 더 생성해야 하는 단점이 있다. 또한 전체 로드 네트워크 중에서 일부의 정보가 변경되었다면 그에 해당하는 NVP 전체를 다시 계산해야 한다. 하지만 우리는 각 노드 종류별로 kNN질의를 처리하기 위해 단순히 각 NVP에 해당 노드 종류의 리스트를 추가 하였다. 이를 통해, 하나의 생성된 NVD로 노드 종류별 kNN질의도 처리 할 수 있게 하였다. 따라서 기존의 연구에 비해, 유지보수 작업이 용이하다.

IV. 질의 처리

우리가 제안하는 시스템은 최근접 질의와 영역

질의를 지원한다. 앞 장에서 살펴보았던 전처리 단계가 끝나면, 질의 응답을 위한 준비가 마무리 되게 된다. 이번 장에서는 실제로 어떻게 질의에 대한 응답이 이루어 지는지 알아 본다.

최근접 질의

최근접 질의는 그 활용이 매우 다양하기 때문에 각종 질의 중에 가장 많이 쓰이는 방법 중에 하나이다. (예제. "내 현재 위치에서 도로의 길이상으로 가장 가까운 병원의 위치를 알려달라.") 우리가 사용할 최근접 질의의 정의는 다음과 같다.

(정의) 전체 도로 망 RN 과 질의 노드 q 가 주어졌을 때, 질의 노드 q 로부터 가장 가까운 k 개의 노드를 찾는다. 단, k 개의 노드는 반드시 모두 q_type 에 속해야 한다. 여기서 가장 가까운 노드라는 것은 질의 노드 q 와 목적지 노드까지의 Euclidean distance 가 아닌 실제 도로 망 상에서의 거리가 가장 가까운 노드이다.

```
Algorithm      kNN( RN, q, q_type, k )
1:   for each node n in RN
2:       g_dist[n] := infinity
3:       p_node[n] := undefined
4:   g_dist[q] := 0
5:   kNN_list := empty
6:   Q := all nodes of RN
7:   while Q is not empty
8:       remove a node u with minimum g_dist
9:       if type(u) = q_type then add a node u to
10:  kNN_list
11:      if size(kNN_list) = k then terminate
12:      for each adjacent connected point v of u
13:          g_new : = g_dist[u] + length(u,v)
14:          if g_new < g_dist[v]
15:              g_dist[v] := g_new
                 p_node[v] := u
```

그림 5. 최 근접 질의 알고리즘(kNN query algorithm)

그림 5는 위의 정의와 같은 질의에 응답하기 위한 알고리즘을 보여준다. 우리가 제안하는 알고리즘은 앞 장에서 살펴본 전처리 과정을 거친 정보를 이용하여 효과적인 질의 처리를 가능하게 한다. 기존의 [7] 방법은 도로 망의 모든 노드가 확장할 노드의 후보에 포함되게 된다. 하지만 우리가 제안하는 방법은 전처리 과정을 통해 탐색공간이 줄어들게 되어, 보다 적은 수의 노드를 방문하는 것으로도 같은 질의에 대한 응답이 가능하다. 이로 인해 탐색 시간이 줄어들게 된다. 특히 해당 q_type의 노드 밀도가 낮은 경우에는 기존의 [7] 방법에 비해 방문 노드 수에서 많은 차이를 보일 수 있으므로 탐색 공간의 감소가 더 큰 효과를 발휘할 수 있다.

우리는 해당 알고리즘의 결과를 통해 다양한 정보를 얻을 수 있다. 먼저 kNN_list 를 통해 실제 어떤 노드가 kNN 노드인지를 알 수 있고, g_dist 와 p_node 를 통해서 질의 노드 q 에서 k 노드로 갈 수 있는 경로와 그 경로의 실제 도로망에서의 길이까지 한번에 파악이 가능하다.

영역 질의

영역 질의 역시 다양한 서비스를 제공하기 위해 널리 사용되는 질의이다. 영역 질의의 예는 다음과 같다: "내 현재 위치에서 도로의 길이상으로 5km 이내에 있는 음식점의 위치를 모두 알려달라." 이번 절에서는 다음과 같이 정의된 영역 질의를 해결하기 위한 알고리즘을 제안한다.

(정의) 전체 도로 망 RN 과 질의 노드 q 가 주어졌을 때, 질의 노드 q 로부터의 거리가 ε 보다 가까운 노드를 모두 찾는다. 단, 여기서 찾아진 모든 노드는 반드시 모두 q_type 에 속해야 한다. 여기서 가장 노드라는 것은 질의 노드 q 와 목적지 노드까지의 Euclidean distance 가 아닌 실제 도로 망 상에서의 거리가 가까운 노드이다.

```
Algorithm      Range( RN, q, q_type,  ε )
1:   for each node n in RN
2:       g_dist[n] := infinity
3:       p_node[n] := undefined
4:   g_dist[q] := 0
5:   Range_list := empty
6:   Q := all nodes of RN
7:   while Q is not empty
8:       remove a node u with minimum g_dist
9:       if g_dist[u] ≤ ε
10:          if type(u) = q_type
```

```
11:            then add a node u to Range_list
12:        else terminate
13:        for each adjacent connected point v of u
14:            g_new : = g_dist[u] + length(u,v)
15:            if g_new < g_dist[v]
16:                g_dist[v] := g_new
17:                p_node[v] := u
```

그림 6. 영역 질의 알고리즘(Range query algorithm)

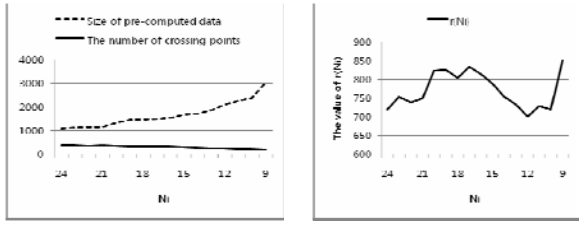
그림 6은 위에 정의된 영역 질의를 위해 제안된 알고리즘이다. 우리는 kNN알고리즘에 간단한 수정을 통해 영역 질의에 대한 응답을 가능하게 하였다. 앞 절에서 살펴본 kNN알고리즘과 유사하게 전처리 과정을 거친 정보를 이용하여 질의에 대한 응답을 한다. g_dist[u]는 질의 노드 q로부터 노드 u까지 이르는 실제 거리가 저장되어 있다. 한편, 해당 알고리즘은 g_dist가 가장 작은 노드를 골라서 노드 u로 선택하게 된다. 따라서 어떠한 노드 u를 방문했다고 했을 때, g_dist[u] 값이 입력된 ε보다 크다면 더 이상 탐색을 하지 않아도 된다. 이러한 과정을 통해 영역 질의에 대한 응답을 효과적으로 할 수 있다.

알고리즘의 실행 결과로, Range_list 를 통해 영역 질의에 맞는 노드들의 목록을 얻을 수 있다. 또한 이 목록과 g_dist, p_node 를 이용하여 질의 노드 q 로부터 해당 노드들까지의 실제 길이와 그 경로를 쉽게 찾아 낼 수 있다.

V. 성능 평가

본 장에서는 제안하는 방법의 성능을 평가하기 위하여 수행한 다양한 실험과 그 결과에 대하여 서술한다. 먼저 전처리 단계에서 어떠한 도로망이 주어졌을 때, 최적의 N_i 를 찾는 함수에 대한 실험을 한다. 그 후에 탐색 시간과 방문 노드 수를 기준으로 제안된 두 가지 알고리즘에 대한 평가를 한다. 알고리즘을 평가할 때, 두 가지 알고리즘에 대한 단일 질의 결과는 그 시간이 매우 짧기 때문에 실험 결과로는 부적합하다. 따라서 우리는 랜덤으로 생성된 500 개의 질의를 하여, 그 결과들의 합을 측정하였다. 본 실험에 사용된 맵 데이터는 972 개의 노드와 4846 개의 엣지로 구성된 랜덤으로 생성된 데이터이다. 실험 환경은 펜티엄 4 3.0Ghz, 2GB RAM, 윈도우 XP SP2 인 PC이다

전처리 단계



a. 필요 저장 공간 및 crossing points 개수(Size of pre-computed data and the number of crossing points)

b. $r(N_i)$

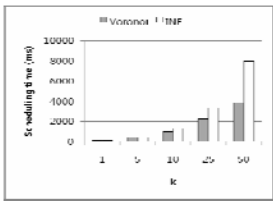
그림 7. 전처리 단계(pre-computation stage)

그림 7-(a)는 N_i 가 변하면서 저장 공간과 crossing points의 개수가 변하는 것을 보여준다. 그림에서 볼 수 있듯이 N_i 가 작아질수록 저장 공간은 선형이상으로 증가 하였고, 반대로 crossing points의 개수는 선형적으로 감소하는 것을 확인할 수 있다. **그림 7**-(b)는 평가함수 $r(N_i)$의 값들을 보여준다. N_i 가 9 정도로 작아지면, 필요한 저장 공간의 크기가 기하급수적으로 늘어난다. 따라서 효율이 좋지 못한 결과가 나왔다. 마찬가지로 N_i 가 15~20 일 때에는 저장 공간의 크기와 탐색 공간의 줄어든 비율이 둘 다 중간 값이기 때문에 효율이 좋지 못한 것으로 나타났다. 이 그래프는 우리가 실험에 사용한 도로망에 대해 N_i 가 12 일 때 NVD가 가장 효율적으로 만들어 진다는 것을 보여준다. 따라서 우리는 앞으로의 실험에서 N_i 를 12 로 하여 진행하였다.

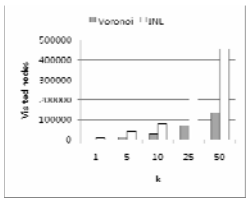
최근접 질의

우리가 제안한 최근접 질의 알고리즘은 비교대상인 INE[7] 알고리즘에 비해, 탐색 시간과 방문 노드 수 모두에서 성능상의 이점을 보여주고 있다. **그림 8**과 **그림 9**는 각각 다른 밀도를 가지고 있는 q_type에 대한 실험 결과를 보여준다. 여기서 밀도는

전체 노드 중에 해당 q_type에 해당되는 노드가 몇 개가 있는지를 나타낸다.



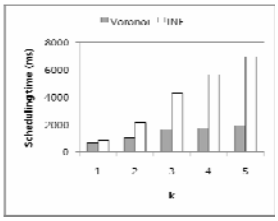
a. 탐색 시간(Scheduling time)



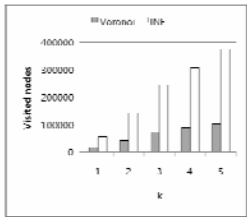
b. 방문 노드 수(Visited nodes)

그림 8. 최근접 질의: 밀도 0.05(kNN query: density 0.05)

그림 8은 밀도가 비교적 높은 q_type에 대한 실험의 결과이다. 이런 경우, k가 작을 때에는 INE과 비교하여 우리가 제안한 알고리즘의 성능은 크게 다르지 않았다. 하지만 더 높은 k에 대한 결과는 우리가 제안한 알고리즘이 훨씬 높은 성능을 보여주는 것을 볼 수 있다.



a. 탐색 시간(Scheduling time)



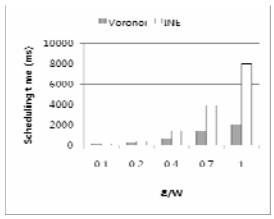
b. 방문 노드 수(Visited nodes)

그림 9. 최근접 질의: 밀도 0.005(kNN query: density 0.005)

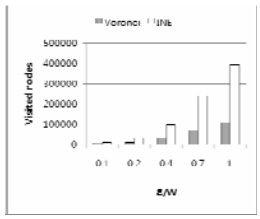
비교적 밀도가 낮은 경우에 대한 실험은 **그림 9**에서 볼 수 있다. 이런 경우에는 밀도가 높은 경우보다 우리가 제안한 알고리즘이 INE에 비교하여 더 좋은 성능을 보이고 있다. 특히 검색 시간에 있어서 그 결과가 더욱 좋게 나왔다. k가 5 일 경우, 밀도가 높을 때에는 INE의 결과와 거의 같은 응답시간이 걸렸다. 하지만 밀도가 낮을 때에는 INE가 걸린 시간의 1/3 도 걸리지 않는 것을 확인했다. 우리가 제안한 최근접 질의 알고리즘은 밀도에 상관없이 INE보다 좋은 성능을 보여주었으며, 특히 밀도가 낮을 때는 더욱 좋은 성능을 보여주었다.

영역 질의

이번 절에서는 우리가 제안한 영역 질의 알고리즘과 RER[7] 알고리즘과 비교해 보았다. 여기서 W는 우리가 사용중인 도로 망의 한 변의 크기이다. **그림 10**은 우리가 제안한 영역 질의 알고리즘에 대한 실험 결과를 보여준다.



a. 탐색 시간(Scheduling time)



b. 방문 노드 수(Visited nodes)

그림 10. 영역 질의(Range query)

앞의 최근접 질의 실험과 마찬가지로, 전체적인 모습은 우리가 제안한 알고리즘의 성능이 뛰어난 것을 확인할 수 있다. 하지만, 이번 경우에 탐색시간에 있어서, ε/W 값이 0.1 일 때는 RER 의 검색 시간이 더 적게 걸리는 결과가 나왔다(141 ms VS. 156 ms). 이것은 두 알고리즘의 구조적인 차이 때문인데, 우리가 제안하는 알고리즘이 전체적으로 방문하는 노드 수는 적지만, 각 노드를 방문했을 때 하는 일은 더 복잡하기 때문에 이러한 결과가 나왔다.

VI. 결론

우리는 본 논문에서 도로 망 데이터베이스를 위한 보로노이 기반 탐색 방법을 제안하였다. 우선 전체 도로망을 여러 개의 NVP 로 나누어서 하나의 NVD 를 생성하고, 이렇게 생성된 각각의 NVP 에 대해 전처리 과정을 수행한다. 여기서 평가함수를 도입하여 이 전처리 과정의 정보가 용량 대비 최대한의 효율을 가지게 하였다. 실제 질의 처리 때에는 전처리 과정을

거친 정보를 이용하여 효과적으로 질의에 대한 응답을 처리할 수 있었다. 또한 기존의 연구에 비해 비교적 간단한 구조를 가지고 있고, 모든 타입의 노드에 대해서 하나의 NVD 를 통해 처리할 수 있기 때문에, 유지보수 면에서도 이점을 가지고 있다. 기존 연구 대비 처리 시간과 방문 노드 수의 두 가지 측면에서 모두 우수함을 보였다.

참고 문헌

[1] C. Hage, C. S. Jensen, T. B. Pedersen, L. Speicys, I. Timko. "Integrated Data Management for Mobile Services in the Real World". VLDB 2003, pp.1019-1030, 2003.

[2] Guttman, A. “R-trees: A Dynamic Index Structure for Spatial Searching”, SIGMOD, 1984.

[3] Sellis, T., Roussopoulos, N. Faloutsos, C. “The R+-tree: a Dynamic Index for Multi-Dimensional Objects”, VLDB, 1987.

[4] Beckmann, N., Kriegel, H.P., Schneider, R., Seeger, B. “The R*-tree: An Efficient and Robust Access Method for Points and Rectangles”, SIGMOD, 1990.

[5] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein. *Introduction to Algorithms, Second Edition*, 2001.

[6] Hjaltason, G., Samet, H. “Distance Browsing in Spatial Databases”, TODS, 24(2), pp. 265-318, 1999.

[7] D. Papadias, J. Zhang, N. Mamoulis, and Y. Tao. “Query Processing in Spatial Network Databases", VLDB, 2003.

[8] M. Kolahdouzan, C. Shahabi. “Voronoi-Based K Nearest Neighbor Search for Spatial Network Databases”, VLDB 2004.

[9] A. Okabe, B. Boots, K. Sugihara, and S. N. Chiu. *Spatial Tessellations, Concepts and Applications of Voronoi Diagrams, 2nd edition*, 2000

[10] F. Aurenhammer. “Voronoi Diagrams - A Survey of a Fundamental Geometric Data Structure”, ACM Computing Surveys, 1991.

[11] D. Lischinski. “Incremental Delaunay triangulation”, Graphics Gems IV, 1994

[12] “LEDA Guide: Duality between Voronoi and Delaunay Diagrams”. http://www.algorithmic-solutions.info /leda_guide/geo_algs/voronoi_delaunay_duality.html.

저 자 소 개

김대훈 (정회원)

2006: 고려대학교 전자공학과 (학사)
2006 ~ 현재: 고려대학교 전자컴퓨터공학과 석사 과정
관심분야: 데이터베이스, 멀티미디어 검색, 영상처리, 유비쿼터스 컴퓨팅
E-mail: kdh812@korea.ac.kr

황인준

1988: 서울대학교 컴퓨터공학과(학사)
1990: 서울대학교 컴퓨터공학과(석사)
1998: Univ. of Maryland at College Park 전산학과 (박사)
1998 ~ 1999: Bowie State Univ., Assistant Professor
1999 ~ 1999: Hughes Research Lab. 연구교수
1999 ~ 2003: 아주대학교 정보통신전문대학원 조교수
2003 ~ 2004: 아주대학교 정보통신전문대학원 부교수
2004 ~ 현재: 고려대학교 전기전자전파공학과 조교수

관심분야 : 데이터베이스, 멀티미디어 검색, 정보 통합, 전자상거래, 영상처리, 유비쿼터스 컴퓨팅
E-mail: ehwang04@korea.ac.kr