보로노이 다이어그램의 경계지점 최소거리 행렬 기반 k-최근접점 탐색 알고리즘

(k-NN Query Processing Algorithm based on the Matrix of Shortest Distances between Border-point of Voronoi Diagram)

엄 정 호* 장 재 우**
(Jung-Ho Um) (Jae-Woo Chang)

요 약 최근 사용자에게 자신과 가장 가까운 k 개의 주유소, 레스토랑, 은행 등의 POI(Point Of Interest) 정보를 추천해주는 위치 기반 서비스가 텔레매틱스, ITS(Intelligent Transport Systems), 키오스크(kiosk)등의 어플리케이션에서 필요로 하고 있다. 이를 위해, 보로노이 다이어그램 k-최근접점 탐색 알고리즘이 제안되었다. 이는 보로노이 다이어그램에서 각 POI의 네트워크의 거리를 미리 계산한 파일을 이용하여 k-최근접점 탐색을 수행한다. 그러나 이 알고리즘은 보로노이 다이어그램 확장에 따른 비용 문제를 야기한다. 따라서 본 논문에서는 보로노이 다이어그램의 경계지점마다 각각에 대하여 최소거리 행렬을 생성하는 알고리즘을 제안한다. 또한 k 개의 POI를 탐색하기 위해, 최소거리 행렬을 이용한 k-최근접점 탐색 알고리즘을 제안한다. 제안하는 알고리즘은 미리 계산된 경계 지점 간 최소거리 행렬을 통해 탐색하므로, k-최근접점 탐색 시 보로노이 다이어그램의 확장비용을 최소화한다. 아울러 기존 연구와의 성능비교를 통해 제안하는 알고리즘이 기존 알고리즘에 비해 검색시간 측면에서 성능이 우수함을 보인다.

키워드 : k-최근접점 탐색 알고리즘, 보로노이 다이어그램, 네트워크 최소거리

Abstract Recently, location-based services which provides k nearest POIs, e.g., gas stations, restaurants and banks, are essential such applications as telematics, ITS(Intelligent Transport Systems) and kiosk. For this, the Voronoi Diagram k-NN(Nearest Neighbor) search algorithm has been proposed. It retrieves k-NNs by using a file storing pre-computed network distances of POIs in Voronoi diagram. However, this algorithm causes the cost problem when expanding a Voronoi diagram. Therefore, in this paper, we propose an algorithm which generates a matrix of the shortest distance between border points of a Voronoi diagram. The shortest distance is measured each border point to all of the rest border points of a Voronoi Diagram. To retrieve desired k nearest POIs, we also propose a k-NN search algorithm using the matrix of the shortest distance. The proposed algorithms can minimize the cost of expanding the Voronoi diagram by accessing the pre-computed matrix of the shortest distances between border points. In addition, we show that the proposed algorithm has better performance in terms of retrieval time, compared with existing works.

Keywords : k-NN search algorithm, Voronoi diagram, network shortest distance

1. 서 론

최근 텔레매틱스, ITS(Intelligent Transport Systems), 키오스크(kiosk)등의 위치 기반 응용 서비스(LBS: Location-Based Service)가 활성화되고 있다. 이러한 LBS 시스템에서 사용자에게 자신과 가장 가까운 몇 개의 주유소, 레스토랑, 은행 등의 POI(Point Of Interest) 정보를 추천해주는 서비스가 필수적이다[1, 2, 4, 5, 6, 7, 8, 9, 10]. 이를 위해 도로 네트워크와 도로 네트워크상에 존재하는 POI를 저장하고, 질의 지점에서 찾고자 하는 POI 개수를 만족할 때까지 네트워크를 확장하는 알고리즘을 사용한다. 이를 k-최근접점(k nearest neighbor: k-NN) 탐색 알고

† 이 논문은 2009년도 정부(교육과학기술부)의 재원으로 한국과학재단의 지원을 받아 수행된 연구임(No. 2009-0059417)

* 전북대학교 전자정보공학부 컴퓨터공학 박사과정, jhum@chonbuk.ac.kr
** 전북대학교 전자정보공학부 컴퓨터공학 교수, jwchang@chonbuk.ac.kr(교신저자)

논문접수 : 2008.05.20 수정일 : 1차 2008.07.23 심사완료 : 2008.07.30

리즘이라 명명한다.

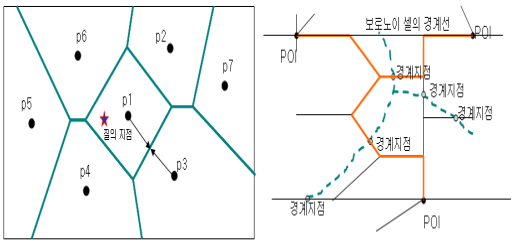
기존 공간 데이터베이스에서는 이러한 k-최근접점을 찾기 위해 질의 지점과 대상 POI와의 직선거리를 고려하여 탐색한다[3]. 하지만 도로와 같은 공간 네트워크상에서 k-최근접점 탐색 알고리즘은 네트워크를 확장해야 하기 때문에, 유클리디언 공간상에서의 최근접점이 네트워크상에서는 최근접점이 아닌 경우가 발생한다[4, 5, 6, 7, 8, 9]. 따라서 k-최근접점 탐색을 위해 네트워크를 확장하면서 질의 지점에서 현재 지점까지의 최소 거리를 유지해야 한다. 이러한 최소거리 유지 방법으로 k-최근접점 탐색 시 네트워크를 확장하는 방법과[4, 5, 6], 네트워크의 거리를 미리 계산하여 네트워크를 확장하는 방법[7, 8, 9] 이 존재한다. 첫째, 네트워크를 확장하는 방법의 대표적인 연구로 홍콩 HKUST 에서 제안한 INE(Incremental Network Expansion)기법이 있다[6]. INE 기법은 네트워크의 에지(edge)를 확장하면서 네트워크의 거리를 계산하여, 사용자로부터 가까운 네트워크 거리 순으로 POI를 탐색하는 기법이다. 이 기법은 에지를 하나씩 확장하므로 이에 대한 비용이 크지만, 네트워크 정보만을 유지하기 때문에 저장 공간을 절약할 수 있는 장점이 있다. 둘째, 네트워크의 거리를 미리 계산하는 방법의 대표적인 연구는 Southern California 대학에서 제안한 보로노이 기반 k-최근접점 탐색 알고리즘(Voronoi-based Network NN Search: VN3)이 있다[7]. 보로노이 기반 k-최근접점 탐색 알고리즘은 각각의 POI가 최근접점이 되는 네트워크 보로노이 다이어그램을 생성한 후, 이를 기반으로 사용자의 POI를 탐색한다. 이 알고리즘은 미리 네트워크 거리를 계산하여 이를 유지하기 때문에, 저장 공간 오버헤드가 크지만 네트워크 확장비용을 크게 감소시킬 수 있는 장점이 있다. 최근 빠른 질의응답 속도에 대한 사용자의 요구가 증가하기 때문에, 네트워크 거리를 미리 계산하는 k-최근접점 알고리즘에 대한 연구가 활발히 진행되고 있다[7, 8, 9]. 이에 대한 연구로 VN3[7], Island 기법[8], Materialization 기법[9]의 연구가 수행되었다. 첫째, Materialization 기법은 네트워크를 구성하는 노드의 개수가 많아지면 저장 공간이 기하급수적으로 증가하는 단점이 있다. 둘째, Island 기법은 POI의 밀도가 증가함에 따라 Island 파일의 각 노드에서 POI까지의 거리를 추가해야 하기 때문에 저장 공간 유지비용이 증가하는 단점이 존재한다. 마지막으로 VN3 방법은 위의 두 알고리즘보다 저장 공간 오버헤드는 적지만, k-최근접점 탐색 알고리즘 수행 시 보로노이 다이어그램 확장에 따른 비용 문제가 존재한다. 따라서 본 논문에서는 이를 해결하기 위하여 보로노이 다이어그램의 경계지점 간 최소거리를 유지함으로써 보로노이 다이어그램을 확장하는 비용을 감소시키는 새로운 k-최근접점 탐색 알고리즘을 제안한다.

본 논문의 구성은 다음과 같다. 2장에서는 기존의 관련 연구를 분석하고, 3장에서는 제안하는 알고리즘을 제시한다. 4장에서는 기존의 연구와 제안하는 알고리즘의 성능 평가를 수행하며, 5장에서는 결론 및 향후연구에 대해 논의한다.

2. 관련 연구

k-최근접 탐색 알고리즘은 크게 네트워크 확장 방법과 미리 네트워크의 거리를 계산하는 방법으로 분류할 수 있다. 대표적인 네트워크 확장 방법으로 INE 기법이 있으며, 네트워크의 거리를 미리 계산하는 방법으로는 VN3, island 기반, Materialization 기반 k-최근접 알고리즘이 존재한다. 첫째, 2003년 홍콩 HKUST 의 D. Papadias et al 은 공간 네트워크를 확장하는 기법인 INE (Incremental Network Expansion)을 제안하였다[6]. INE 기법은 질의가 속한 에지를 탐색하여, 이 에지를 시작으로 에지 상에 존재하는 k개의 최근접점을 찾을 때까지 네트워크를 확장해가는 방법이다. 이를 통해 얻어지는 k-최근접 POI들은 공간 네트워크상의 거리에 기반하여 계산된다. 확장을 수행할 때, 새로이 추가되는 에지까지의 거리가 k번째 최근접 네트워크 거리보다 커지면 알고리즘은 종료된다. 그러나 INE 알고리즘은 기본적으로 Dijkstra 알고리즘을 사용하여 최근접점을 찾기 때문에, 지도 데이터에 비해 POI 밀도가 낮은 경우 전체적인 검색 시간이 증가한다.

둘째, 2004년 Southern California 대학의 M. Kolahdouzan et.al 은 유클리디언 환경에서 연구되어진 보로노이 다이어그램을 네트워크 데이터베이스에 적용한 k-최근접점 탐색 방법(VN3)을 제안하였다[7].



(a) 보로노이 다이어그램 (b) 네트워크 보로노이 다이어그램

그림 1. 보로노이 다이어그램

보로노이 다이어그램이란 유클리디언 공간상에 많은 정점들이 있을 때 각각의 정점에서의 같은 거리에 떨어진 지점들을 기준으로 공간을 분할하는 방법이다(그림 1-(a)). 보로노이 다이어그램은 질의 지점이 P1의 셀안에 존재할 때, P1이 질의 지점의 가장 가까운 정점이라는 것을 보장한다. 이를 도로 네트워크에 적용할 경우에는 그림 1의 (b)와 같다. 기존의 보로노이 다이어그램과 달리 네트워크에 보로노이 다이어그램을 적용할 때에는 각각의 보로노이 셀간의 거리를 구하기 위해 최소거리 계산 알고리즘인 Dijstra 알고리즘을 사용한다. 네트워크 보로노이 다이어그램을 사용한 보로노이 기반 k-최근접 질의처리 알고리즘은 다음과 같다. 첫째, 질의점이 포함되어 있는 보로노이 영역(Network Voronoi Polygon)을 검색함으로써, 가장 가까운 POI을 찾는다. 둘째, 최근접점 POI

가 포함되어 있는 영역(질의영역)의 이웃 영역에서 두 번째 POI를 찾을 수 있으므로, 이웃하는 영역을 검색한다. 이 때, 이웃 영역에 속해있는 POI까지의 실제거리를 미리 계산하여, 계산된 보로노이 셀의 경계지점에서 POI까지의 거리, 경계지점에서 경계지점까지의 거리를 이용하여 두 번째 최근접점을 구한다. 아울러 질의점이 속해 있는 영역과 두 번째 최근접점이 포함된 영역을 합쳐 하나의 질의영역으로 만들고, 해당영역의 경계지점까지의 거리를 재계산한다. 위와 같은 방식으로 이웃하는 영역에서 k번째 최근접점을 탐색할 때까지 보로노이 셀을 확장하는 방법을 사용한다. 하지만 이러한 방법은 POI가 밀집되어 있을 경우, 확장 횟수가 증가하여 검색의 효율성이 감소한다.

셋째, 2005년 노르웨이 Alborge 대학의 X. Huang 는 POI에서 노드까지의 거리를 유지하는 Island 기반 k-최근접점 탐색 알고리즘을 제안하였다[8]. Island 기법에서는 POI에서 전체 네트워크의 노드까지의 거리를 유지하는 것은 큰 저장 오버헤드를 가지므로 범위(Radius)를 설정하여 저장 비용을 감소시킨다. 아울러, 네트워크의 에지를 아이디와 에지의 길이로, POI는 에지의 시작노드에서부터의 offset 정보로 표현한다. Island 기반 k-최근접 알고리즘은 다음과 같이 동작한다. 첫째, 질의 지점을 포함하는 에지를 구성하는 노드를 힙(Heap)에 저장한다. 힙은 질의 지점에서 노드까지의 거리 순으로 저장된다. 둘째, 질의 지점에서 거리가 가장 가까운 노드를 확장한다. 셋째, 확장한 노드의 Island 파일을 검색하여 파일에 있는 POI 정보를 검색한다. 마지막으로, 만일 POI 개수가 k개를 만족하지 못하면, k를 만족할 때까지 반복한다. Island 기법은 범위를 넓게 설정하여 각 노드로부터 모든 POI들의 거리를 계산하면 k개의 POI를 검색하는 시간이 적게 소요되지만, 저장 비용이 기하급수적으로 증가하는 문제점을 지닌다. 또한, POI 갱신 시 모든 노드에 이를 반영해야 하기 때문에, 갱신 비용이 커지는 문제점이 존재한다.

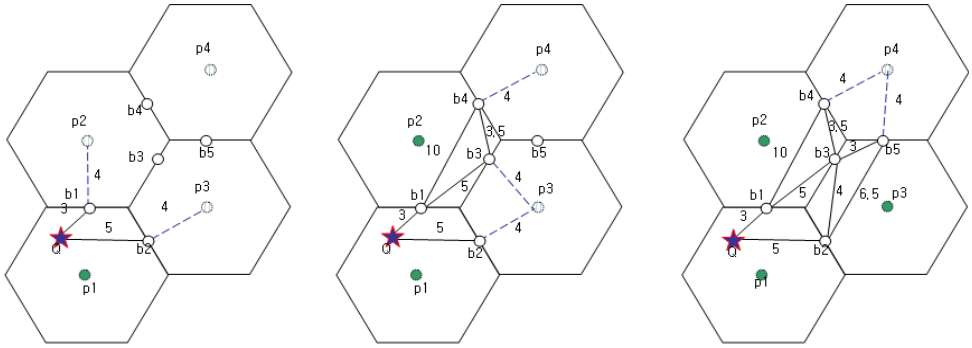
마지막으로 2007년 전북대학교에서는 Materialization 기반 k-최근접점 탐색 알고리즘을 제안하였다[9]. 이 기법은 먼저 질의 처리를 위해 Materialization 파일을 생성한다. Materialization 파일은 네트워크의 모든 노드에서 노드까지의 거리를 계산하여 질의 지점이 주어지면 이를 포함하는 두 노드에서 다른 노드까지의 거리가 저장된 Materialization 파일을 이용하여 질의를 수행한다. 이 알고리즘은 노드간의 거리를 미리 계산하여 저장하는 Materialization 파일을 사용하여 디스크 I/O 및 검색에 필요한 노드간의 거리계산을 감소시킨다. Materialization 기반 k-최근접점 탐색 알고리즘은 다음과 같이 동작한다. 첫째, 질의 지점을 포함하는 에지에 위치에 있는 POI를 탐색하여 후보 결과 집합에 저장한다. 둘째, 질의 지점이 포함되어 있는 에지의 두 노드로부터 다른 모든 노드까지의 거리와 순서 정보를 Materialization 파일로부터 획득한다. 셋째, 불러온 Materialization 파일의 정보를 이용하여 질의 지점에서 가장 가까운 노드부터 k가 될 때까지 확장을 반복한다. 마지막으로, 후보 결과 집합을 정렬한 후, 상위 k개를 최종 결과 집합으로 반환한다. 이 기법은 k-최근접점 탐색 처리에 있어서 POI의 밀도에 상관없이 우수한 성능을 유지할 수 있으나, 노드간의 최소거리를 모두 유지하기 때문에, 디스크의 저장 오버헤드가 커지는 문제점이 존재한다.

3. 보로노이 다이어그램의 경계지점 최소거리 행렬 기반 k-최근접점 탐색 알고리즘

최근 제안되는 공간 네트워크상에서 k-최근접점 탐색 알고리즘은 미리 네트워크 거리를 계산하여 k-최근접점을 탐색하는 방법을 사용한다. 이에 대한 연구로 VN3, Materialization 기법과 Island 기법이 수행되었다. 그러나 Materialization 기법은 네트워크를 구성하는 노드의 개수가 많아지면 저장 공간이 기하급수적으로 증가하므로 이에 대한 유지 관리 비용이 크다. 또한, Island 기법은 POI의 밀도가 증가함에 따라 Island 파일의 각 노드에 POI까지의 거리를 추가해야 하기 때문에 저장 공간 유지 비용이 증가한다. VN3 기법은 위의 두 알고리즘보다 저장 공간 오버헤드는 적지만 k-최근접점 탐색 알고리즘 수행 시 보로노이 다이어그램 확장에 따른 비용 문제가 존재한다. 이를 해결하기 위해 보로노이 다이어그램의 경계지점 간 최소거리를 유지함으로써, 보로노이 다이어그램을 확장하는 비용을 감소시키는 방법이 필요하다. 아울러 이러한 최소거리를 유지하는 저장 비용은 네트워크 노드간 최소거리를 유지하는 저장비용보다 효율적이어야 한다. 따라서 본 논문에서는 보로노이 다이어그램의 경계지점 간 최소거리를 유지하고, 이를 통해 k-최근접점을 탐색하는 알고리즘을 제안한다.

3.1 기존 보로노이 k-최근접점 탐색 알고리즘의 문제점

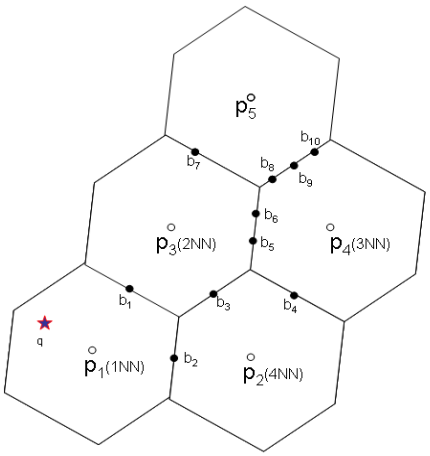
기존의 보로노이 다이어그램 기반 k-최근접 질의 처리 알고리즘[7]은 보로노이 셀 내부의 경계지점 간 거리만을 유지한다. 따라서 k개의 최근접점을 찾기 위해서는 질의 지점에서 해당하는 보로노이 셀까지의 최소거리를 계산해야 한다. 이를 위해 보로노이 셀의 경계지점들을 노드로 하는 네트워크를 구성하고 최소거리 계산 알고리즘을 통해 거리를 계산한다. 이 계산과정은 기존의 네트워크 확장 방법과 동일하기 때문에 거리의 갱신 문제가 존재한다. 그림 2는 기존 보로노이의 k-최근접점 수행과정을 나타낸다. 그림 2의 (a)에서 P1은 질의지점이 있는 셀이므로 첫 번째 POI로 선택되고, 두 번째 POI를 찾기 위해서 Q에서 다른 POI인 P2와 P3까지의 거리를 경계지점에서 POI까지 미리 계산된 거리를 통해 계산한다. 그림 2의 (b)는 세 번째 POI를 검색하기 위해 P1과 P2에 해당하는 보로노이 셀의 경계지점들 간의 네트워크를 구성하여 이웃 POI인 P4와 P3까지의 거리를 계산한다. 그림 2의 (c)는 네 번째 POI를 검색하는 과정이며, 여기에서 그림 2의 (b)에서 설정한 P4까지의 최소 거리 15.5 (Q-b1-b3-b4-P4 구간)가 최소거리 구간이 Q-b1-b3-



(a)두 번째 POI 검색 (b) 세 번째 POI 검색 (c)네 번째 POI 검색

그림 2. 기존 보로노이 k-최근접점 탐색 수행과정

b5-P4 으로 변경되면서 Q에서 P4까지의 최소거리 또한 15 로 갱신된다. 그림 3은 보로노이 k-최근접점 탐색 수행 시, 거리 계산 과정을 보여준다. 그림 3의 (a)는 k-최근접점 탐색을 수행하는 보로노이 다이어그램을, (b)는 질의 수행 시 거리 계산 구간과 구간 계산 시 추가적으로 필요한 거리 계산 횟수를 보여준다. (b)의 테이블의 첫 번째 열은 현재 검색한 POI를 나타낸다. 두 번째 열은 각각의 최근접 POI를 검색하기 위해 계산해야 하는 구간과 구간의 거리를 계산하기 위해 새로 계산해야 하는 구간의 수를 나타낸다. 세 번째 열은 POI를 찾기 위해 필요한 총 계산 횟수를 보여준다. 그림 3의 예제에서 k개의 POI 최근접점의 순서는 P1-P3-P4-P2순이다, P1은 질의 지점을 포함하는 보로노이 셀이므로 첫 번째 최근접점이 되고, 두 번째 POI를 찾기 위해 이루어지는 구간 계산은 2회이다. P3 은 두 번째 POI 이고 이때의 구간 계산은 4회 이루어진다. P4 는 세 번째 POI 이고, 네 번째 POI를 찾기 위해 셀을 확장할 때 구간 b5-b8-b9-b10 와 b6-b8-b9-b10 을 계산한다. 이는 최소거리가 두 보로노이 셀을 경유하기 때문이다. 이와 같이 질의지점에서 후보 셀을 계산하기 위해, 두 셀을 경유하는 경우에는 최소거리 계산을 위해, 두 셀의 모든 경계지점을 지나는 경우를 고려하여 거리를 비교해야 한다. 그림 4는 이를 나타낸다. 그림 3에서 b8, b9, b10 은 그림 3의 (a)에서 POI P5와 P4간의 경계지점이다. 그림 4에서 n1~n7까지는 셀에 해당하는 POI 인 P4에 포함되는 노드이고, n8~n13까지는 P5에 포함되는 노드이다. 만일 POI P4의 경계지점 b6에서 P5까지의 최소거리를 계산한다면, b6-b8-P5, b6-b9-P5, b6-b10-P5의 경우를 고려해야 한다. 즉, 각각의 최소거리를 그림 4의 테이블을 이용하여 계산하면, b6-b8-P5 는 30, b6-b9-P5 는 54, b6-b10-P5 는 38이다. b6-b8-P5 의 거리가 제일 짧기 때문에, b6-P5까지의 최소거리는 30이 된다. 그러나 그림 4의 네트워크를 살펴보면, b6-b8-b9-b10-P5 의 거리가 28 로 실제 최소거리가 됨을 알 수 있다. 이러한 상황 때

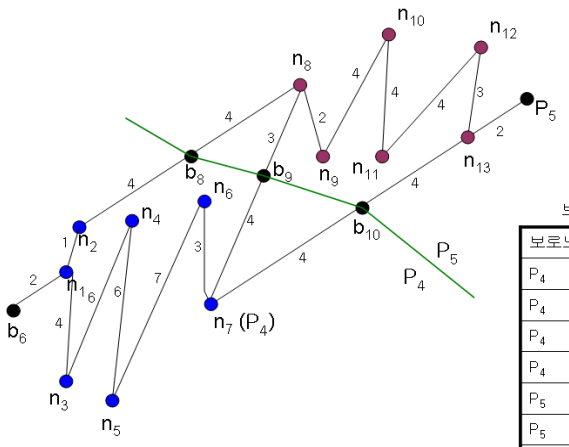


(a)네트워크 보르노이 다이어그램

확장 POI	계산하는 구간 (새로 계산하는 횟수)	총 계산 횟수
P_1	$q-b_1(1)$, $q-b_2(1)$	2
P_3	$b_1-b_7(1)$, $b_1-b_3(1)$, $b_1-b_5(1)$,$b_1-b_6(1)$	4
P_4	$b_5-b_8(1)$, $b_5-b_9(1)$, $b_5-b_{10}(1)$, $b_5-b_8-b_9-b_{10}(3)$, $b_6-b_8(1)$, $b_6-b_9(1)$, $b_6-b_{10}(1)$, $b_6-b_8-b_9-b_{10}(3)$	12
P_2	$b_2-b_3-b_5-b_8(2)$, $b_2-b_3-b_5-b_9(2)$, $b_2-b_3-b_5-b_{10}(2)$, $b_2-b_3-b_5-b_8-b_9-b_{10}(2)$, $b_2-b_3-b_6-b_8(2)$, $b_2-b_3-b_6-b_9(2)$, $b_2-b_3-b_6-b_{10}(2)$, $b_2-b_3-b_6-b_8-b_9-b_{10}(2)$, $b_2-b_4-b_8(2)$, $b_2-b_4-b_9(2)$, $b_2-b_4-b_{10}(2)$, $b_2-b_4-b_8-b_9-b_{10}(2)$, $b_1-b_3-b_4-b_8(2)$, $b_1-b_3-b_4-b_9(2)$, $b_1-b_3-b_4-b_{10}(2)$, $b_1-b_3-b_4-b_8-b_9-b_{10}(2)$	32
계산 횟수 합계		50

(b)k-최근접점 질의 수행시 거리 계산 테이블

그림 3. 보로노이 k-최근접점 질의 수행시 거리 계산 과정



보르노이 셀 최소거리 테이블

보로노이 셀	구간	최소거리
P_4	b_6-b_8	7
P_4	b_6-b_9	32
P_4	b_6-b_{10}	32
P_4	b_9-b_{10}	8
P_5	b_8-P_5	23
P_5	b_9-P_5	22
P_5	b_{10}-P_5	6
P_5	b_8-b_9	7

그림 4. 최소거리 계산시 두 보로노이 셀을 경유하는 예제

문에 POI P4를 확장할 때 b8, b9, b10의 거리 조합으로 가능한 모든 경우를 고려하여 최소거리를 계산한다. 그림 3의 (b)에서 마지막 POI 는 P2이다. (b)에서 보는 것과 같이 2개의 새로 계산되는 구간이 있다. 이는 P2를 확장하면서 새로 삽입된 구간 b2-b3을 계산해야 하고, 이로 인하여 b3-b5 와 b3-b6 를 재계산해야 하기 때문이다. 아울러, b2-b3, b2-b4, b3-b4 구간 삽입으로 인하여, 질의 지점에서 b8, b9, b10 의 경계지점까지 최소거리를 갱신해야 한다. 이와 같은 경우를 일반화하면 (그림 5)과 같다.

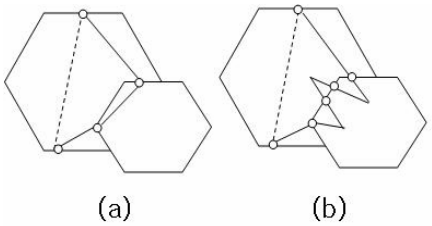


그림 5. 보로노이 k-최근접점 탐색 알고리즘 수행 시 갱신이 발생하는 경우

그림 5는 보로노이 k-최근접점 탐색 알고리즘 수행 시 POI를 확장할 때 갱신이 발생하는 경우를 보여준다. 그림 5의 (a)는 그림 4에서 P2 와 같은 경우를 나타내며, 그림 5의 (b)는 P4 와 같은 경우를 나타낸다. 이와 같이 보로노이 k-최근접점 질의 처리 알고리즘은 확장할 때마다 항상 재계산을 수행하며, 이는 보로노이 k-최근접점이 지역적으로 거리를 확장하기 때문이다. 즉, 질의 지점에서 POI의 거리까지의 최소거리를 계산하는 것은 최소 거리 알고리즘을 수행하는 시간과 동일하고, m을 보로노이 셀에서 경계지점간의 구간 총 개수, n을 모든 경계지점의 개수라 할 때 k-최근접점 탐색을 위해 소요되는 시간은 $k\Theta(m+nlgn)$와 같다. 따라서 이와 같이 소요되는 시간을 감소시킬 필요가 있다.

3.2 경계지점 최소거리 행렬 생성 알고리즘

k-최근접점 탐색 알고리즘을 위한 저장구조는 크게 다음의 세 가지 정보를 저장한다. 첫째, 보로노이 셀 정보는 이웃하는 보로노이 셀 식별자, 셀의 영역, 보로노이 셀의 경계 지점을 저장한다. 둘째, 셀 내부 최소거리 정보는 셀 내부의 POI와 노드 간의 최소거리, POI와 경계간의 최소거리를 저장한다. 셋째, 경계지점 최소거리 행렬은 보로노이 셀의 경계지점간의 최소거리를 저장한다. 저장하는 정보 중에서 보로노이 셀 정보와 셀 내부 최소거리 정보는 기존의 보로노이 셀 생성방법과 동일하다. 한편 경계지점 최소거리 행렬의 생성방법은 다음과 같다. 먼저 최소거리를 계산할 두 경계 지점을 bi(1≤i≤n)와 bj(1≤j≤n)라 하고, 경계지점을 포함하는 노드를 n(i,j)(1≤i≤n, j=1|2)라 할 때, 최소거리는 다음의 수식을 통해 정의된다. 먼저, 정의 1은 임의의 두 경계지점간의 가능한 4가지 경우를 정의한다. 정의 2는 경계지점간의 최소거리를 정의한다.

정의 1. 경계지점간의 가능한 최소거리

D1,1=Dist(n(i,1),bi)+Dist(n(i,1)+n(j,1))+Dist(n(j,1),bj)
D1,2=Dist(n(i,1),bi)+Dist(n(i,1)+n(j,2))+Dist(n(j,2),bj)
D2,1=Dist(n(i,2),bi)+Dist(n(i,2)+n(j,1))+Dist(n(j,1),bj)
D2,2=Dist(n(i,2),bi)+Dist(n(i,2)+n(j,2))+Dist(n(j,2),bj)
단, i≠j

정의 2. 경계지점간의 최소거리

MinDisti,j=Min(D1,1 , D1,2 , D2,1 , D2,2)

그림 6은 경계지점간의 가능한 4가지 경우를 계산하는 과정을 보여준다. 그림 6과 같이 bi는 n(i,1),n(i,2) 사이에 있고, bj는 n(j,1),n(j,2) 사이에 있다. 이 때 가능한 최소거리 경우의 수는 D1,1 , D1,2 , D2,1 , D2,2 와 같고 이 중에서 제일 최소가 되는 거리가 MinDist로 설정된다.

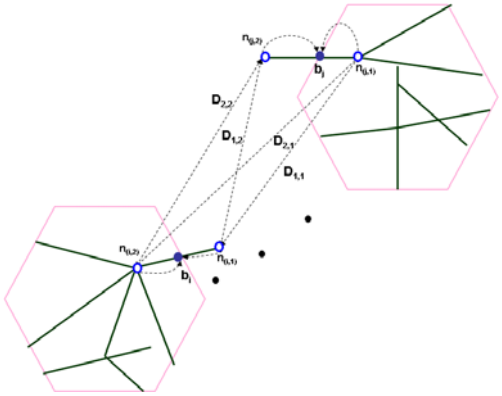


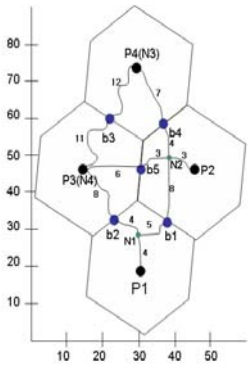
그림 6. 경계 지점 최소거리 행렬 생성 과정

경계 지점 최소거리 행렬의 생성 알고리즘은 먼저, 경계 지점 최소거리 행렬을 생성하기 위해 최소거리 행렬을 생성할 두 경계지점 bi와 bj 를 검색한다(단계 1-3). 경계지점이 검색되면 해당 경계 지점을 포함하는 에지를 검색하고 최소거리를 저장할 변수인 MinDist 를 초기화한다(단계 4-5). bi와 bj의 두 경계지점을 포함하는 에지의 각 노드에서 경계지점까지의 거리를 계산하여 이 중에서 가장 최소가 되는 거리를 찾는다. 최소거리는 MinDist에 저장된다(단계 6-9). 최소거리 MinDist가 결정되면 이를 최종적으로 최소거리 행렬에 저장한다. 알고리즘은 그림 7과 같다.

Algorithm GenerateBorderToBorderShortestPathMatrix
Input: 모든 경계지점
Output: 경계 지점 간 최소거리 행렬
1.For each bi{ //bi 는 임의의 경계지점
2. For each bj{// bj 는 bi와 다른 경계지점
3. Search bi and bj border-point
4. Search edge including bi and bj
5. MinDist=Double Max
6. For each Ni // Ni는 bi를 포함하는 노드 집합
7. if(MinDist > Dist(Ni,Nj)+Dist(Ni,bi))
then MinDist=Dist(Ni,Nj)+Dist(Ni,bi)
8. For each Nj // Nj는 bj를 포함하는 노드 집합
9. if(MinDist > Dist(Ni,Nj)+Dist(Nj,bj))
then MinDist=Dist(Ni,Nj)+Dist(Nj,bj)
10. Store the MinDist to bi's array and bj's array
11.}}//end of for
End Algorithm

그림 7. 경계 지점 최소거리 행렬 생성 알고리즘

그림 8은 경계 지점 최소거리 행렬의 예를 보여준다. 그림 8에서 POI 는 4개이고, POI 간의 네트워크 보로노이 다이어그램을 계산하여 경계지점을 설정한 결과 경계지점은 b1~b5까지 5개가 생성된다. POI의 P3 와 P4 는 노드 N3 와 N4 에 위치한다. 보로노이 셀 정보 테이블에는 P1~P4까지의 보로노이 셀 정보가 저장되고, 셀 내부 최소거리 정보 테이블에는 각 POI에서 경계지점까지의 최소거리와 셀에 포함된 노드에서 경계지점까지의 최소거리가 저장된다. 그림에서는 P1의 최소거리 정보를 보여준다. 경계지점 최소거리 정보 테이블에는 모든 경계지점에서 경계지점까지의 거리를 저장한다. 예제에서는 경계지점이 5개 존재하므로 5*5의 최소거리를 저장하는 행렬을 그림 8에서와 같이 생성한다.



경계지점 최소거리 정보 테이블

	b1	b2	b3	b4	b5
b1	0	9	27	12	11
b2	9	0	19	21	14
b3	27	19	0	19	17
b4	12	21	19	0	7
b5	11	14	17	7	0

그림 8 경계 지점 최소거리 행렬을 포함한 저장 구조의 예

3.3 k-최근접 탐색 알고리즘

k-최근접점 탐색을 수행하기 위해서는 먼저 질의 지점에서 경계지점까지의 거리를 계산해야 한다. 하지만 질의 지점을 포함하는 에지의 노드가 서로 다른 보로노이 셀을 가질 수 있기 때문에, 두 노드에서 경계지점까지의 거리를 이용하여 최소거리를 계산한다. 정의 3은 이를 고려하여 최소거리를 선택하는 기준을 정의한다.

정의 3. 질의 지점에서 경계지점까지의 최소거리
Dist(q,bi)=Min(Dist(q,n1)+Dist(n1,bi),
Dist(q,n2)+Dist(n2,bi))
질의 지점에서 POI까지의 최소거리는 질의 지점에서 경계지점까지의 거리, 경계 지점에서 POI의 경계지점까지의 거리, 그리고 POI의 경계지점에서 POI까지의 거리를 합산한 거리이다. 이는 정의 4 와 같다.

정의 4. 질의 지점에서 POI까지의 최소거리
MinDistq,p=Min(Dist(q,bi)+Dist(bi,bj)+Dist(bj,P)),
0≤i≤n1, 0≤j≤n2,
n1=질의 지점이 포함된 보로노이 셀의 경계지점 개수
n2=확장하는 POI의 보로노이 셀의 경계지점 개수
여기서 Dist(q,bi), Dist(bi,bj), Dist(bj,P)를 계산할 때,

셀 내부 최소거리 정보와 경계지점 최소 거리 행렬 저장 구조에서 미리 계산된 거리를 검색한다. 아울러, Dist (bi,bj)의 경우 기존 보로노이 k-최근접점 탐색 방법에서는 지역적으로 거리를 확장하기 때문에, 확장하는POI 셀까지의 거리를 재계산하지 않으면 최소거리를 계산할 수 없었다. 그러나 제안하는 알고리즘은 경계지점간의 최소거리를 계산할 때 미리 계산된 경계지점 최소거리 행렬을 이용하여 최소거리를 유지하기 때문에 거리를 재계산 없이 검색할 수 있다. 즉, 최소거리가 배열로 저장되어, 바로 접근 가능하기 때문에 경계지점의 수에 관계없이 검색시간은 일정하다. 따라서 k개의 POI를 검색할 때 거리 계산하는 시간은$\Theta(kc)$(c : 계산된 거리의 검색 시간)가 된다.

한편 정의 3과 정의 4에서 정의한 보로노이 다이어그램의 경계지점 최소 거리 행렬을 이용한 k-최근접점 탐색 알고리즘은 다음과 같다. 먼저 질의 지점을 포함하는 보로노이 셀을 검색한다. 검색 후, POI개수가 k와 같은지 확인한다. 만일 k와 같지 않다면, 질의 지점을 포함하는 에지가 두 개의 셀에 걸쳐 있는지를 확인한다. 만일 그렇다면, 두 개의 셀을 모두 고려하여, 정의 3을 이용하여 질의 지점으로부터 경계지점까지의 거리를 계산한다. 아울러 질의 지점과 이웃하는 보로노이 셀을 대상으로 질의 지점에서 POI까지의 최소 거리를 계산한다. 최소거리를 계산하면, 디스크에 있는 경계지점 최소 거리 행렬에서 질의 지점을 포함하는 셀의 경계지점의 최소거리 열을 검색하여 메모리에 적재한다. 이 과정의 효율성을 위해, 경계지점의 최소거리 행렬을 생성할 때 경계지점의 열을 단위로 하여 저장한다. 이웃 보로노이 셀을 검색한 후, 메모리에 적재된 경계지점간의 최소거리 행렬을 이용하여 정의 4에 따라 최소거리를 검색한다. 최소거리 검색 후, 미리 계산된 셀 내부의 최소거리 파일에서 경계지점에서 POI까지의 거리를 검색한다. 이 거리를 최소 힙(Min Heap)에 삽입한다. 최소 힙을 사용하여 다음 최근접점을 얻어와 k가 될 때까지 앞의 과정을 반복한다. (그림 9)는 k-최근접점 탐색 알고리즘을 보여준다.

```
Algorithm Global Voronoi Network diagram based k-Nearest Neighbor Search
Input: k, query point
Output: Pois
1. Search Voronoi Cell(qV)
   which containing query point
2. if k==1 return
3. else if(query point is ling on two voronoi cell){
     3.1. Compute qtob distance
          considering two voronoi cell
4.}else Compute qtob distance
5. Search BSA(Border Shortest path Array)with qV
6. while(Pois !=k){
        6.1 Search Adjacency Voronoi Cell
        6.2 Compute qtob distance by BSA
        6.3 Insert qtob distance into MinHeap
        6.4 Extract MinHeap
        6.5. gotoNextPoi
7. }//of while
End Algorithm
```

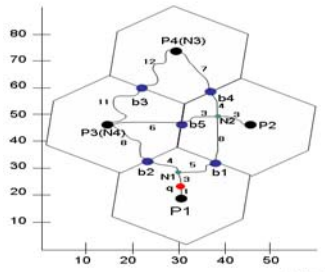
그림 9. 보로노이 다이어그램 기반 최소거리 유지 k-최근접점 탐색 알고리즘

제안하는 알고리즘의 예제는 그림 10과 같다. 그림에서 q는 질의 지점을 나타내며 P1에서의 거리는 1이고, N1에서의 거리는 3이다. 예를 들어 3개의 최근접점을 찾는다면 그림의 k-최근접점 탐색 단계 테이블에서의 순서로 탐색을 수행한다. 단계 1에서 P1이 첫 번째 최근접점으로 검색되며, P1의 셀 내부 최소거리 정보 테이블에서 질의지점이 있는 N1에서 경계지점까지의 거리를 검색한다. b1 까지 거리는 5이고, b2는 4 이다. 단계 2 에서 P1의 인접 POI는 P2와 P3 이므로, P2와 P3까지 거리를 셀 내부 최소거리 정보 테이블에서 계산하면 b1에서 11 b2에서는 8이 된다. 정의 4 에 따라 P3 가 두 번째 최근접점으로 선택된다. 단계 3에서 P3의 인접한 셀은 P2와 P4이고 P2와 연결된 경계지점은 b5이고 P4와 연결된 경계지점은 b3이므로 b1-b3, b1-b5, b2-b3, b2-b5의 거리를 경계지점 최소거리 정보 테이블에서 계산한다. 정의 4 에 따라 거리 19 를 가지는 P2가 세 번째 최근접점으로 결정된다. 단계 4에서 최근접점의 결과는 P1, P3, P2 가 선택된다.

그림 11은 기존 보로노이 최근접점 탐색수행시의 거리 계산과 제안하는 k-최근접점 탐색 수행 시 거리 계산 테이블을 보여준다. 그림 11의 (b)와 (c)에서 보는 것과 같이 P1과 P3 는 거리 계산 횟수가 같다. 그러나 P4 의 경우 12번에서 6번으로 6번의 계산횟수가 줄어들고, P2의 경우 32에서 0번으로 계산횟수가 줄어든다. P4 의 경우, 기존 보로노이 k-최근접점 탐색 알고리즘에서는 P4에서 P5까지의 최소거리를 계산하기 위해, P4와 P5의 경계지점을 순회하여 지나가는 모든 경우의 수를 계산한다. 그러나 제안하는 알고리즘은 질의 보로노이 셀의 P1의 경계지점에서 P5 까지의 최소거리를 경계지점 최소거리 열을 통하여 알 수 있기 때문에, P4 의 경우 6번의 계산 연산이 감소된다. 아울러, P2 의 경우 기존 알고리즘에서는 P2가 4-최근접점으로 선택됨에 따라, 질의 지점에서 후보 집합까지의 최소거리를 재계산한다. 그러나 제안하는 알고리즘은 전체 네트워크상의 최소거리를 유지하기 때문에, P2 의 선택으로 인한 갱신 연산이 불필요하다. 따라서 제안하는 알고리즘에서 P2의 경우 계산 횟수가 0번이다.

4. 성능평가

성능평가는 본 연구에서 제시한 k-최근접 질의 처리 알고리즘과 관련연구의 INE 기법, VN3, Island(IL), Materialization(MAT) 기법을 비교한다. 제안하는 알고리즘은 GVN(Global Voronoi Network diagram)라 명명한다. Island 알고리즘의 경우 범위는 전체 네트워크의 10%로 한다. 성능평가 환경은 Intel Xeon CPU 3GHz, 메모리 2GByte이고, 운영체제는 Window2003에서 Visual C로 구현하였다. 성능평가에 사용한 데이터는 샌프란시스코만 데이터[11]이며, 노드는 약 17만개, 에지는 약 22만개이다. POI는 10,000개를 사용하였다. VN3와 GVN에서 사용하는 R*-tree의 페이지 크기는 1KB를 사



보로노이 셀 정보 테이블

POI	인접 POI	MBR (x1,y1,x2,y2)	경계지점
P1	P2, P3	(15,2,42,35)	b1,b2
P2	P3, P4	(32,28,58,62)	b1, b4, b5
P3	P1, P4	(1,25,30,65)	b2,b3,b5
P4	P2, P3	(18,52,45,88)	b3,b4

셀 내부 최소거리 정보 테이블

POI or Node	경계지점	최소거리
P1	b1	8
P1	b2	9
N1	b1	5
N1	b2	4
...	...	...
...	...	...

경계지점 최소거리 정보 테이블

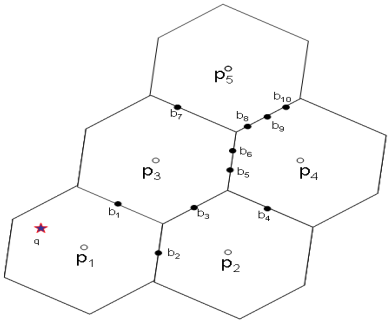
	b1	b2	b3	b4	b5
b1	0	9	27	12	11
b2	9	0	19	21	14
b3	27	19	0	19	17
b4	12	21	19	0	7
b5	11	14	17	7	0

(a) 저장 구조

단계	결과 POI	후보 POI (ID(거리)	계산 구간 (구간-구간(거리))
1		P1(1)	q-b1(8), q-b2(7)
2	P1	P2(19), P3(15)	b1-p2(11), b2-p3(8)
3	P1, P3	P2(19), P4(38)	b1-b3(27), b1-b5(11), b2-b3(19), b2-b5(14), b3-p4(12), b5-p2(6)
4	P1,P2,P3		

(b) k-최근접점 탐색 수행 단계

그림 10. 보로노이 다이어그램 기반 최소거리 유지 k-최근접점 탐색 알고리즘의 예



(a)네트워크 보르노이 다이어그램

확장 POI	계산하는 구간 (구간 거리 계산 횟수)	총 계산 횟수
P_1	$q-b_1(1)$, $q-b_2(1)$	2
P_3	$b_1-b_7(1)$, $b_1-b_3(1)$, $b_1-b_5(1)$,$b_1-b_6(1)$	4
P_4	$b_5-b_6(1)$, $b_5-b_9(1)$, $b_5-b_{10}(1)$, $b_5-b_8-b_9-b_{10}(3)$, $b_6-b_8(1)$, $b_8-b_9(1)$, $b_6-b_{10}(1)$, $b_6-b_8-b_9-b_{10}(3)$	12
P_2	$b_2-b_3-b_5-b_6(2)$, $b_2-b_3-b_5-b_9(2)$, $b_2-b_3-b_5-b_{10}(2)$, $b_2-b_3-b_5-b_9-b_{10}(2)$, $b_2-b_3-b_6-b_8(2)$, $b_2-b_3-b_5-b_9(2)$, $b_2-b_3-b_6-b_{10}(2)$, $b_2-b_3-b_5-b_8-b_9-b_{10}(2)$, $b_2-b_4-b_8(2)$, $b_2-b_4-b_9(2)$, $b_2-b_4-b_{10}(2)$, $b_2-b_4-b_8-b_9-b_{10}(2)$, $b_1-b_3-b_4-b_8(2)$, $b_1-b_3-b_4-b_9(2)$, $b_1-b_3-b_4-b_{10}(2)$, $b_1-b_3-b_4-b_8-b_9-b_{10}(2)$	32
계산 횟수 합계		50

(b)기존의 보로노이 다이어그램 기반 k-최근접점 탐색 알고리즘

확장 POI	계산하는 구간 (구간 거리 계산 횟수)	총 계산 횟수
P_1	$q-b_1(1)$, $q-b_2(1)$	2
P_3	$b_1-b_7(1)$, $b_1-b_3(1)$, $b_1-b_5(1)$,$b_1-b_6(1)$	4
P_4	$b_1-b_8(1)$, $b_1-b_9(1)$,$b_1-b_{10}(1)$, $b_2-b_8(1)$, $b_2-b_9(1)$,$b_2-b_{10}(1)$	6
P_2		0
계산 횟수 합계		12

(c)제안하는 k-최근접점 탐색 알고리즘

그림 11. 제안하는 알고리즘의 k-최근접점 탐색 수행 시 거리 계산테이블

용하였으며, GVN의 경우 질의 처리 알고리즘 수행 시 질의가 포함된 셀의 경계지점 최소거리 행렬은 메모리에 유지한다.

4.1 k-최근접 질의 처리 알고리즘 수행 시간

본 절에서는 k-최근접 질의 처리 알고리즘의 수행 시간을 평가한다. 실험에 사용한 k의 개수는 10, 20, 50, 100이다. 그림 12는 k-최근접 질의 처리 알고리즘의 수행 시간 성능평가를 나타낸다.

그림 12에서 보는 것과 같이 k가 10일 때는 검색 시간 순으로 VN3가 0.01초, INE가 0.02초, GVN이 0.02초, MAT가 0.04초, IL이 0.1초가 소요된다. k가 10일 때는 VN3와 INE가 MAT와 IL보다 좋은 성능을 보인다. 이는 POI를 탐색하기 위해 네트워크를 확장하는 구간이 적

기 때문에, 우수한 성능을 나타낸다. GVN은 INE와 유사한 성능을 보이지만 타 기법보다는 우수한 검색 성능을 보인다. k가 20일 때는 검색 시간 순으로 VN3가 0.02초, GVN이 0.03초, INE가 0.04초, MAT가 0.05초, IL이 0.06초가 소요된다. k가 20일 때는 제안하는 알고리즘인 GVN이 INE보다 우수한 성능을 보인다. 이는 GVN의 최소거리 계산 비용이 네트워크 확장비용보다 적어짐을 의미한다. k 가 50일 때는 검색 시간 순으로 GVN이 0.04초, VN3가 0.06초, IL이 0.06초, MAT가 0.08초, INE가 0.15초가 소요된다. 이와 같이 k가 50이상이 되면 INE, VN3의 알고리즘보다 IL과 GVN의 검색 시간이 더 적어짐을 알 수 있다. 이는 k가 커짐에 따라 네트워크를 확장하는 비용이 최소거리 파일을 탐색하는 비용보다 더 크기 때문이다. GVN과 IL중에서는 GVN의 검색 시간이 더 적게 소요된다. 그 이유는 IL은 일정 범위(실험에서는 10%)의 네트워크 정보만을 유지하므로 이를 초과할 때 네트워크의 확장이 필요한 반면, GVN은 보로노이 다이어그램의 경계 지점 간 최소 거리를 유지하므로 이러한 확장이 불필요하기 때문이다.

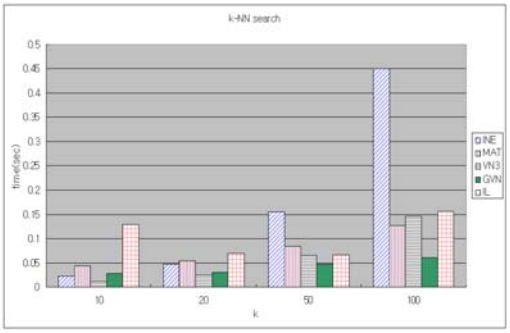


그림 12. k-최근접 질의 처리 알고리즘 수행 시간 성능평가

4.2 저장 공간 오버헤드

본 절에서는 각 k-최근접 질의 처리 알고리즘에서 미리 계산된 네트워크 거리의 저장 공간 오버헤드를 살펴본다. 표 1은 저장 공간 오버헤드를 나타낸다. 각 저장 공간 오버헤드는 최소거리를 유지하는 알고리즘만을 대상으로 하였기 때문에, INE 기법은 비교대상에서 제외하였다. 표 1에서 VN3와 GVN의 8MB는 보로노이 다이어그램을 유지하기 위한 저장 공간이며, 나머지는 미리 계산된 네트워크 거리를 위한 저장 공간이다. 저장 공간 오버헤드는 VN3 알고리즘이 가장 적은 오버헤드를 지니며, GVN, IL, MAT기법 순으로 더 큰 오버헤드를 지닌다. IL 기법의 경우 네트워크 정보의 10%를 최소거리 파일로 유지하므로 GVN 보다 적은 저장 공간 오버헤드를 지니지만, 이 비율을 증가시킬 경우 GVN 보다 큰 저장 공간을 요구한다. 따라서 본 논문에서 제안하는 알고리즘은 중간 정도의 성능을 나타낸다. 즉, MAT 기법보다는 우수한 성능을 보이고, IL 이나 VN3 기법보다는 저하된 성능을 나타낸다.

표 1. 저장 공간 오버헤드

알고리즘	필요 저장공간
MAT	346GB
VN3	263MB+8MB
IL	6GB
GVN	10GB+8MB

5. 결 론

공간 네트워크상의 k-최근접 질의 처리 알고리즘은 크게 네트워크 확장 방법과 미리 계산된 네트워크 거리를 유지하는 방법으로 나눌 수 있다. 이 중에서 본 연구에서는 미리 계산된 네트워크 거리를 유지하는 방법에 초점을 맞추어, 보로노이 다이어그램의 최소거리를 유지하는 행렬을 이용한 k-최근접 질의 처리 알고리즘을 제안하였다. 제안하는 알고리즘은 두 가지의 장점을 가진다. 첫째, 최소거리 행렬을 유지함으로써 네트워크의 확장비용을 상수의 시간으로 감소시켰고, 성능평가를 통해 기존의 보로노이 k-최근접점 탐색 알고리즘보다 효율적임을 보였다. 둘째, 기존의 Materialization 기법과 같이 큰 저장 공간의 오버헤드를 가지지 않으면서도 빠른 k-최근접점 탐색 시간을 달성하는 알고리즘을 제안하였다. 아울러 각 기법에 대한 실제 저장 공간을 비교하여, 제안하는 알고리즘이 저장 공간 측면에서도 성능이 크게 저하되지 않음을 보였다. 향후 연구로는 본 논문에서 제안한 알고리즘을 확장하여 연속 k-최근접점 탐색 알고리즘을 설계하는 것이다. 아울러 제안하는 알고리즘을 다양한 응용 서비스에 적용하여 실세계에 적용할 수 있음을 보이는 것이다.

참 고 문 헌

[1] S. Shekhar et al., “Spatial Databases Accomplishments and Research Needs,” IEEE Tran. on Knowledge and Data Engineering, Vol. 11, No. 1, 1999, pp. 45-55.

[2] L. Speicys, C.S. Jensen, and A. Kligys, “Computational Data Modeling for Network-Constrained Moving Objects,” Proc. of ACM GIS, 2003, pp. 118-125.

[3] N. Roussopoulos, S. Kelley and F. Vincent, “Nearest neighbor queries,” Proc. of ACM SIGMOD,1995, pp. 71-79.

[4] C.S. Jensen, J. Kolar, T.B. Pedersen, and I. Timko, “Nearest Neighbor Queries in Road Networks,” Proc. of ACM GIS, 2003, pp. 1-8.

[5] C. Shahabi, M.R. Kolahdouzan, M. Sharifzadeh, “A Road Network Embedding Technique for

K-Nearest Neighbor Search in Moving Object Databases," GeoInformatica, Vol. 7, No. 3, 2003, pp. 255-273.

[6] D. Papadias, J. Zhang, N. Mamoulis, and Y. Tao, "Query Processing in Spatial Network Databases" Proc. of VLDB, 2003, pp, 802-813.

[7] M. Kolahdouzan and C. Shahabi, "Voronoi-Based K Nearest Neighbor Search for Spatial Network Databases", Proc. of VLDB, 2004, pp. 840-851.

[8] X. Huang, C. S. Jensen, S.Saltenis. "The Islands Approach to Nearest Neighbor Querying in. Spatial Networks." In Proc. SSTD, 2005, pp. 73 - 90.

[9] 김용기, 니하드 카림 초우더리, 이현조, 장재우, "공간 네트워크 데이터베이스에서 실체화 기법을 이용한 범위 및 k-최근접 질의처리 알고리즘", 한국공간정보시스템학회 논문지, 제 9권 1호, 2007, pp. 67~79.

[10] 김용기, 김아름, 장재우, "공간 네트워크 데이터베이스에서 공간 제약을 고려한 경로 내 최근접 질의처리 알고리즘", 한국공간정보시스템학회 논문지, 제 10권 3호, 2008, pp. 19-30.

[11] T. Brinkhoff, "A Framework for Generating Network-Based Moving Objects," In Proc. of GeoInformatica 6(2), 2002, pp. 153-180.

엄 정 호

2004년 전북대학교 컴퓨터공학과 (공학사)
2004년~2006년 전북대학교 컴퓨터공학과 (공학석사)
2006년~현재 전북대학교 컴퓨터공학과 박사과정
관심분야는 공간 데이터베이스, 공간색인 구조, GIS

장 재 우

1984년 서울대학교 전자계산기공학과 (공학사)
1986년 한국과학기술원 전산학과 (공학석사)
1991년 한국과학기술원 전산학과 (공학박사)
1996년~1997년 Univ. of Minnesota, Visiting Scholar
2003년~2004년 Penn State Univ., Visiting Scholar.
1991년~현재 전북대학교 컴퓨터공학과 교수
관심분야는 공간 네트워크 데이터베이스, 상황인식, 하부저장구조