공간 네트워크 데이터베이스에서 시간 및 공간제약을 고려한 In-Route Nearest Neighbor 질의처리 알고리즘 설계

*김상미°, *장재우
*전북대학교 컴퓨터 공학과
{smkim°, jwchang}@dblab.chonbuk.ac.kr

Design of In-Route Nearest Neighbor Query Processing Algorithm with Time and Space-constraint in Spatial Network Databases

*Sang-mi Kim°, *Jae-woo Chang
*Dept. of Computer Engineering, Chonbuk National University

요 약

최근 공간 네트워크 데이터베이스를 위한 질의처리 알고리즘에 관한 연구가 많이 진행되어 왔다. 그러나 현재 좌표-기반 질의에 대한 연구는 활발히 진행중인 반면, 경로-기반 질의에 대한 연구는 매우 미흡한 실정이다. 공간 네트워크 데이터베이스에서는 이동객체가 공간 네트워크상에서만 이동하기 때문에 경로-기반 질의의 유용성이 매우 증대되므로, 경로-기반 질의에 대한 효율적인 질의처리 알고리즘 연구가 필수적이다. 따라서 본 논문에서는 경로-기반 질의의 대표적인 방법인 In-Route Nearest Neighbor 질의처리 알고리즘을 분석하여 기존 연구에서 고려하지 않은 시간 및 공간제약을 고려한 경로-기반 질의처리 알고리즘을 설계한다.

1. 서 론

최근 공간 네트워크 데이터베이스를 위한 질의처리 알고리즘에 관한 연구(좌표-기반 질의, 궤적-기반 질의, 경로-기반 질의)가 활발히 진행되어 왔다. 그러나 현재 좌표-기반 질의에 대한 연구는 활발히 진행 중이나, 경로-기반질의에 대한 연구는 매우 미흡한 실정이다. 공간 네트워크 데이터베이스에서는 이동객체가 공간 네트워크 상에서만 이동하기 때문에 경로에 기반한 질의의 유용성이 매우 증대되며, 따라서 경로-기반 질의에 대한 효율적인 질의처리 알고리즘 연구가 필수적이다. 미국 Minnesota대학의 연구에서는 대표적인 경로-기반 질의로써 In-Route Nearest Neighbor(이하 IRNN으로 명명) 질의를 정의하였으며[1], 이는 경로를 가장 적게 벗어나면서 최근접점을 찾는 데에 초점을 맞추고 있다. 그러나 IRNN 질의처리 알고리즘은 첫째, 시간제약을 고려하지 못하기 때문에 시간제약 범위안의 POI(Point of Interest)를 검색하지 못하는 문제점이 발생한다. 둘째, 공간 제약을 고려하지 못하기 때문에 병목현상이 발생한 도로 내의 실시간 상황을 반영하지 못하는 문제점이 존재한다. 이러한 문제점을 극복하기 위해서 본 논문에서는 기존 IRNN 질의처리 알고리즘을 기반으로, 시간 및 공간제약을 고려한 IRNN 질의처리 알고리즘을 설계한다.

본 논문의 구성은 다음과 같다. 2장에서는 관련 연구를 소개하고, 3장에서는 본 논문에서 설계한 시간 및 공간 제약을 고려한 IRNN 질의처리 알고리즘을 제시한다. 마지막으로 4장에서는 결론 및 향후연구를 제시한다.

2. 관련 연구

기존의 경로-기반 질의는 연속 최근접(continuous Nearest Neighbor)질의, 연속 k-최근접(continuous k-Nearest Neighbor)질의, IRNN(In-Route Nearest Neighbor)질의 등으로 구성된다. 일반적으로 이들 경로-기반 질의들의 공통점은 이동 객체가 이동하면서 최근접점 POI를 탐색한다는 점이다. 아울러 이들 경로-기반 질의는 유클리디언 공간에서의 연속 k-최근접(연속 최근접 포함) 질의와, 공간 네트워크에서의 연속 k-최근접 질의 및 IRNN질의로 분류할 수 있다. 상대적으로 유클리디언 공간에서의 연속 k-최근접 질의에 대한 연구가 다수 진행된 데 반해, 공간 네트워크에서의 연속 k-최근접 질의 및 IRNN질의에 대한 연구는 초보적인 단계에 있다.

첫째, 미국 Illinois 대학에서는 유클리디언 공간에서 연속 최근접 질의에 대해 처음으로 정의하였다[2]. 아울러 미국 Maryland대학에서는 fixed upper bound algorithm을 통해 이동객체가 새로운 최근접 질의를 수행하지 않

고 움직일 수 있는 최소 거리를 정의하여, 연속 최근접 질의를 처리하기 위한 알고리즘을 제시하였다[3].

둘째, 중국 HKUST에서는 time parameterized 질의의 개념에 기초하여 연속 최근접 질의를 처리하는 알고리즘을 제시하였다[4].아울러 HKUST에서는 유클리디언 공간에서 객체의 전체 경로에 대해 하나의 최근접점 질의로 근사화 시키는 방법을 통해 연속 k-최근접 질의처리 알고리즘을 제시하였다[5].

셋째, 미국 Southern California대학에서는 공간 네트워크상에서 연속 k-최근접 질의를 처리하기 위한 알고리즘을 제시하였다[6]. 이 알고리즘에서는 Voronoi network diagram을 통해 미리 찾고자 하는 POI들의 네트워크 거리를 미리 계산하고, 이동객체의 경로가 주어지면 각각의 노드별로, 또는 경로 상에 POI가 있으면 노드와 POI로 경로를 세그먼트로 분할한다. 각각의 세그먼트별로 K개의 POI를 Voronoi diagram으로 찾은 다음, POI의 방향성에 따라 세그먼트의 분할 점을 설정하여 그 분할 점에서의 K개의 후보 셋을 만든다. 이와 같은 방법은 성능이 비효율적이므로 Maryland 대학에서 제시한 upper bound algorithm을 사용하여 분할 점의 개수를 줄이는 방법을 제시하였다.

마지막으로 Minnesota대학의 연구에서는 IRNN질의를 제시하였다[1]. 연속 최근접 또는 연속 k-최근접 질의가 가장 짧은 경로로 갈 수 있는 최근접점을 찾는데 초점을 맞추었다면, IRNN질의는 경로를 가장 적게 벗어나면서 최근접점을 찾는 데에 초점을 맞춘다. 이를 위해 4가지 알고리즘을 제안하였다. 첫째, 단순 그래프(Simple graph) 기반 방법으로, 경로상의 각 노드마다 최근접점을 찾은 후 각 노드의 최근접점을 경유하여 목적지에 도달하는 최단 거리를 계산하여 가장 최단거리가 되는 최근접점을 찾는 방법이다. 이 방법은 노드가 많아지면 비용이 증가한다는 단점이 있다. 둘째, *recursive spatial* range질의 기반 방법으로, 현재 경로상의 최근접점을 최단거리 알고리즘(shortest path algorithm)을 통해 찾은 후 , 다음 노드에서의 최근접점을 맨 처음에 찾은 최근접점까지의 거리로 영역 질의를 통해 찾는다. 만일 여기서 찾은 최근접점의 거리가 더 작다면 영역 범위를 이 거리로 조절한 후 다음 노드에서 영역질의를 수행한다. 이 방법은 첫 번째 방법에 비해 비용이 감소하나 각 노드에서의 영역질의를 수행하는 비용이 많이 든다. 셋째, spatial distance 조인 기반 방법으로, 각 노드에서의 영역질의를 수행하는 비용을 줄이기 위해 제안되었다. 이 방법은 노드와 POI에 대한 두 가지의 트리를 가지고 경로 상에 존재하는 각 노드와 POI의 유클리디언 거리 조인을 통해 영역질의 범위를 구하여 최근접점을 구한다. 마지막으로, precomputed zone 기반 방법은 각 POI마다 서비스 지역(service zone)을 설정하여 노드에서 POI까지의 네트워크 거리를 미리 계산한다. 따라서 이동객체의 경로가 주어지면 해당경로를 포함하는 서비스 지역을 검색하여 최근접점을 빠르게 찾을 수 있다. 이상 제안된 4가지 방법 가운데 precomputed zone 기반 방법이 가장 좋은 성능을 보이고 있다.

3. 시간 및 공간제약을 고려한 IRNN질의처리 알고리즘

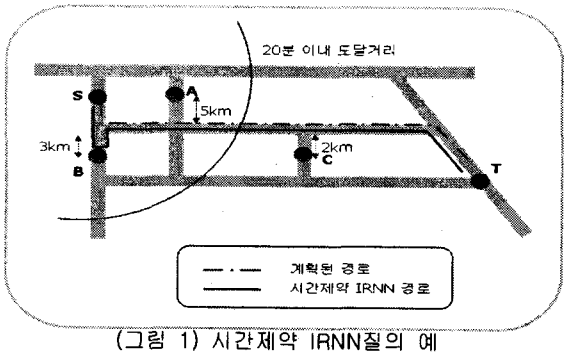
시간 및 공간제약을 고려한 IRNN질의처리 알고리즘 설계의 목적은 텔레매틱스(telematics), CNS(Car Navigation System), 자동항법장치, L-commerce 등과 같은 실제 응용에서 시간 및 공간제약을 이용하여 최적 조건의 POI 및 경로를 탐색하는 것이다.

이를 위하여 경로-기반 질의의 대표적인 방법인 미국 Minnesota대학의 연구에서 제시한 IRNN질의처리 알고리즘에서 가장 좋은 성능을 갖는 precomputed zone 방법을 기반으로, 시간 및 공간제약을 고려한 IRNN 질의처리 알고리즘을 설계한다.

3.1 시간제약 IRNN질의처리 알고리즘

기존 IRNN질의는 설정된 경로를 가장 적게 벗어나면서 방문할 수 있는 POI를 검색하는 질의이며, POI검색 시 시간상의 어떠한 제약도 존재하지 않는다. 하지만, 실제 텔레매틱스나 L-commerce 응용에서는 시간제약이 중요한 역할을 수행한다. 일반적으로 자동차가 예정된 경로를 운행할 때, 현재위치에서 원하는 시간 안에 찾고자하는 POI가 있을 수 있다. 이 경우에는 기존 IRNN질의에서 찾은 POI가아닌 원하는 시간 안에 POI를 검색해야한다. (그림1)을 통해 예를 들면, "현재 지점(S)에서 시청(T)까지 최단 경로를 탐색하고 그 경로를 운행하고자 할 때, 설정된 경로를 가장 적게 벗어나면서 현재 지점(S)에서 20분 내에 방문할 수 있는 주유소를 검색하라"와 같은 질의이다.

(그림1)에서 A, B, C로 표시된 것은 주유소 POI를 나타내며, 기존 IRNN질의처리 알고리즘은 계획된 경로를 가장 적게 벗어나면서 최근접점을 찾는 데에 초점을 맞추기 때문에 주유소 C를 검색한다. 하지만, 만약 현재 자동차가 20분 내에 급유를 해야 운행이 가능하다면, 경로를 가장 적게 벗어나면서 20분이라는 시간제약 범위 내에 존재하는 주유소 B를 검색해야한다. 따라서 이러한



(그림 1) 시간제약 IRNN질의 예

응용을 지원하기 위해 기존 IRNN질의에 시간제약 개념을 고려한 시간제약 IRNN질의처리 알고리즘의 개발이 요구된다. 한편, 시간제약 개념을 고려한 IRNN질의처리 알고리즘의 설계 시, 시간제약 정도의 강약에 따라 POI를 검색하는 연구가 필요하다. 앞선 예의 경우, 주어진 시간 내에 급유하지 않으면 더 이상 운행이 불가능하므로 시간제약 정도가 매우 강하지만, 식당의 경우는 시간제약 정도가 상대적으로 강하지 않다. 즉, 검색대상 POI의 성격에 따라 시간제약 정도에 가중치를 두고, 비용함수를 통해 계산하여 최적 조건의 POI를 찾는 질의처리 알고리즘에 대한 연구가 필수적이다. 이를 위해 검색하고자 하는 대상 POI의 시간제약 정도를 사용자로부터 입력받아 최적 조건의 POI를 경유하는 경로비용함수(route cost function)를 다음과 같이 정의한다.

$$Cost_{IRNN} = (MinDist_{POI} \times 2 + Dist_{IRRoute})/Dist_{IRRoute}$$
$$Cost_{TCRoute} = (MinDist_{TCPOI} \times 2 + Dist_{IRRoute})/Dist_{IRRoute}$$
$$Cost_{TCIRNN} = MIN((1-\alpha) \times Cost_{TCRoute}, \alpha \times Cost_{IRNN}) \text{--(1)}$$

식 (1)에서 $Cost_{TCIRNN}$은 기존IRNN경로 및 시간제약을 고려한 IRNN경로 가운데 적은 비용을 가진 경로를 반환한다. 여기서, α는 사용자로부터 입력받은 검색 대상 POI에 대한 시간제약 가중치이다. 먼저, 시간제약 범위 내에 이동할 수 있는 거리는 (속력 x 시간제약범위)/α로 계산하여 구한다. 여기서 $Cost_{TCRoute}$는 시간제약 범위내의 최적 조건의 POI를 경유하는 경로를 포함한 경로비용을 나타낸다. $Cost_{IRNN}$는 시간제약을 갖지 않는 기존 IRNN질의처리 알고리즘에서의 POI를 경유하는 경로비용을 나타낸다. 또한, MinDist는 주어진 경로의 한 노드에서 검색대상 POI까지의 최단거리, $Dist_{IRRoute}$는 IRNN경로의 총 운행거리이다. 즉, (그림1)을통해 설명하면, $Cost_{TCRoute}$는 20분 이내에 갈수 있는 POI중 B를 경유하여 목적지까지 가는 경로의 비용을 의미하며, $Cost_{IRNN}$는 기존IRNN 질의처리 알고리즘에서 찾은 주유소 C를 경유하여 목적지까지 가는 경로의 비용을 의미한다. 또한, MinDist는 각각의 주어진 경로에서 가장 가까운 POI와의 거리인 3km, 5km, 2km이며, $Dist_{IRRoute}$는 출발지(S)에서 목적지(T)까지의 계획된 경로의 총 운행거리이다. 이러한 경로비용함수를 바탕으로 설계한 시간제약 IRNN질의처리 알고리즘은 그림(2)와 같다. 첫째, 사용자에 의해 지정된 경로내의 노드를 B+-트리에서 검색하여 각 노드마다 가장 가까운 POI를 테이블에 저장한다. 둘째, 사용자의 현재위치에서 시간제약 범위 내에 이동할 수 있는 거리를 (이동거리=속력 x 시간제약)/α공식을 통해 구한다.
셋째, 테이블로부터 경로에서 가장 가까운 POI와 시간제약 범위 내에서 경로에서 가장 가까운 POI를 검색한다. 넷째, 각각 검색된 POI를 경유하는 경로인 TCRoute와 IRNN Route를 설정한다. 마지막으로, 사용자로부터 입력받은 시간제약에 대한 가중치 α가 1인 경우에는 시간제약 범위 내의 POI를, 0인 경우에는 경로에서 가장 가까운 POI를 전달한다. 아울러 $1>\alpha>0$인 경우에는 α를 반영한 TCRoute와 IRNN Route에 대한 경로비용을 계산하고, 보다 적은 경로비용을 지닌 경로의 POI를 전달한다.

시간제약 IRNN (BaseRoute, POI, Time, α)
//BaseRoute: 기준경로, POI: 검색대상, Time:시간제약, α:시간제약 가중치

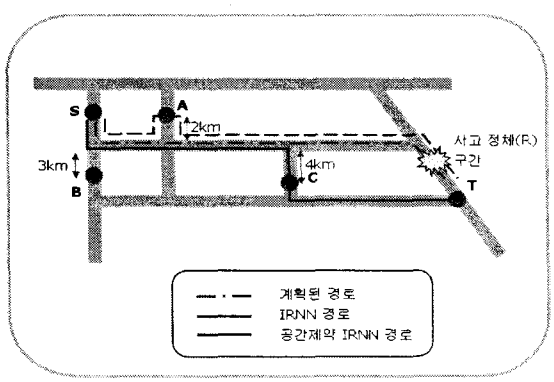
```
1. Result=Load_B+Tree(BaseRoute);

2. 시간제약 범위 내 이동거리 = (속력 x 시간제약)/α;

3. TCPOI=findTimeConstrainedPOI
          (시간제약 범위 내 이동거리, Result);

4. POI=IRNN(Result);
5. if( α==1 ) return TCPOI;
6. else if( α==0 ) return POI;
7. else
  {
    Dist_Route =Compute_Distance(BaseRoute);
    Cost_IRNN = (MinDist_POI x 2 +
                    Dist_IRRoute)/Dist_IRRoute;
    Cost_TCRoute = (MinDist_TCPOI x 2 +
                    Dist_IRRoute)/Dist_IRRoute;
    return MIN((1-α)×Cost_TCRoute , α×Cost_IRNN);
  }
```

(그림 2) 시간제약 IRNN질의처리 알고리즘

이러한 시간제약 IRNN질의처리 알고리즘은 기존의 IRNN이 시간제약을 고려하지 못하기 때문에 시간제약 범위안의 POI를 검색하지 못하는 문제점을 해결할 수 있다. 그러나 기존 IRNN질의처리 알고리즘에서 설정된 경로를 가장 적게 벗어나면서 방문할 수 있는 POI를 검색하는데 반해, 시간제약 IRNN질의처리 알고리즘은 시간제약 범위안에 POI는 검색할 수 있지만 설정된 경로에서 가장 적게 벗어나면서 방문할 수 있는 POI가 아닌 경우가 있을 수 있다.

3.2 공간제약 IRNN질의처리 알고리즘

기존 IRNN질의처리 알고리즘은 질의를 만족하는 POI 검색 시, 시간상의 제약뿐 아니라 공간상의 제약도 고려하지 않는다. 그러나 실제 텔레매틱스나 L-commerce 응용에서는 시간뿐만 아니라 공간제약도 중요한 역할을 담당한다. 일반적으로 자동차가 예정된 경로를 운행할 때, 설정된 경로에 교통사고나 교통체증이 발생할 경우 사용자가 그 구간을 지나지 않고 새로운 길로 가기를 원할 수 있다. (그림3)을 통해 예를 들면, "현재 지점(S)에서 시청(T)까지 평소 운행하는 경로에서, 사고가 발생하여 정체되는 R구간을 통하지 않고 은행을 경유하는 새로운 경로(평소 운행하는 경로를 가장 적게 벗어나는)를 찾아라"와 같은 질의이다.



(그림 3) 공간제약 IRNN질의 예

(그림3)에서 A, B, C로 표시된 것은 은행 POI를 나타내며, 기존 IRNN질의처리 알고리즘은 계획된 경로를 가장 적게 벗어나면서 최근접점을 찾는 데에 초점을 맞추기 때문에 은행A를 경유하는 점선으로 표시된 경로를 선택한다. 그러나 계획된 경로 중 사고가 발생하여 정체되는 구간(R구간)이 있다면, 이 구간(R구간)은 비용이 많이 소요되기 때문에 이를 제외시키고, R 구간을 지나지 않고 평소 운행하는 경로를 가장 적게 벗어나는 새로운 경로(실선으로 표시된 경로)를 선택해야한다. 또한, 공간제약을 고려한 IRNN질의처리 알고리즘의 설계 시, 공간제약 정도의 강약에 따른 경로 선택 연구가 필요하다. (그림3)에서는 사용자가 해당 사고구간(R구간)을 경유하지 않고 다른 경로로 은행(POI)을 방문하였지만, 해당 구간에 반드시 방문해야할 POI가 존재하여 사용자가 해당 구간을 꼭 경유해야 하는 경우도 발생한다. 따라서 사용자의 선호도나 선택에 따라 공간제약 정도에 가중치를 두고, 비용함수를 통해 최적 조건의 경로를 탐색하는 질의처리 알고리즘에 대한 연구가 필수적이다. 이를 위해 사용자로부터 공간제약 정도에 대한 가중치를 입력받아 최적조건의 경로를 탐색하는 경로비용함수(route cost function)를 다음과 같이 정의한다.

$$Cost_{IRNN} = (MinDist_{POI} \times 2 + Dist_{IRRoute})/Dist_{IRRoute}$$

$$Cost_{SCRoute} = (MinDist_{SCPOI} \times 2 + Dist_{IRRoute} + (Dist_{IRRoute} - Dist_{Route}))/Dist_{IRRoute}$$

$$Cost_{SCIRNN} = MIN((1-\beta) \times Cost_{SCRoute}, \beta \times Cost_{IRNN}) \text{--(2)}$$

여기서 MinDist는 주어진 경로의 한 노드에서 검색대상 POI까지의 최단거리, $Dist_{IRRoute}$는 IRNN경로의 총 운행거리, $Dist_{Route}$는 공간제약을 고려하여 기존 경로를 우회하는 경로의 총 운행거리이며, β는 사용자로부터 입력받은 공간제약 가중치이다.

식(2)에서 $Cost_{SCIRNN}$은 기존IRNN경로 및 공간제약을 고려한 IRNN경로 가운데 적은비용을 가진 경로를 반환한다. $Cost_{SCRoute}$는 공간제약을 고려하여 최적조건의 경로로 설정된 경로비용을 나타내며, $Cost_{IRRoute}$는 공간제약을 갖지 않는 기존 IRNN질의처리 알고리즘에서의 경로비용을 나타낸다. 한편, $Cost_{SCRoute}$비용을 구하기 위해 SCRoute(정체되는 R구간을 제외한 새로운 경로)를 설정할 때, 공간제약구간을 지나지 않고 평소 운행하는 경로를 가장 적게 벗어나는 최적의 새로운 경로를 구하기 위해 모든 경로(Route)에 다음과 같은 공식을 적용하여 가장 적은 값을 갖는 경로의 비용을 $Cost_{SCRoute}$로 설정한다.

$$Cost_{SCRoute} = MIN\{ (MinDist_{SCPOI} \times 2 + Dist_{IRRoute} + (Dist_{IRRoute} - Dist_{Route}))/Dist_{IRRoute} \text{ , for all Route}\}$$

이를 위해 사용자로부터 공간제약의 정도에 대한 가중치

를 입력받아 최적 조건의 경로를 탐색하는 공간제약 IRNN질의처리 알고리즘은 그림(4)와 같다. 첫째, 공간제약 구간(R구간)의 노드 쌍을 설정한다. 둘째, 공간제약구간을 지나지 않고 평소 운행하는 경로를 가장 적게 벗어나는 최적의 새로운 경로 SCRoute를 구한다. 셋째, β가 1인 경우에는 공간제약 구간을 포함하지 않는 경로인 SCRoute의 SCPOI를 반환한다. 넷째, β가 0인 경우에는 공간제약을 고려하지 않는 의미이기 때문에 기존 IRNN경로의 POI를 반환한다. 아울러 $1>\beta>0$인 경우에는 β를 반영한 SCRoute와 IRNN Route에 대한 경로비용을 계산하고, 보다 적은 경로비용을 지닌 경로의 POI를 전달한다.

공간제약 IRNN (BaseRoute, POI, Link, β)
//BaseRoute: 기준경로, POI: 검색대상, Link:공간제약 구간, β:공간제약 가중치

1. SCRoute=infinite;
2. $Dist_{IRRoute}$ =Compute_Distance(BaseRoute);
3. for (All routes)
   3.1 SCDist = ($MinDist_{SCPOI} \times 2 + Dist_{IRRoute} + (Dist_{IRRoute} - Dist_{Route})) / Dist_{IRRoute}$;
   3.2 if(SCRoute>SCDist)
       SCRoute=SCDist;
4. if(β==1)
   4.1 SCRoute=findSpaceConstrainedPOI(SCRoute);
   4.2 return SCPOI;
5. else if(β==0)
   5.1 SCRoute=IRNN(BaseRoute);
   5.2 return POI;
6. else
   {
   $Cost_{IRNN} = (MinDist_{POI} \times 2 + Dist_{IRRoute}) / Dist_{IRRoute}$;
   $Cost_{SCRoute} = (MinDist_{SCPOI} \times 2 + Dist_{IRRoute} + (Dist_{IRRoute} - Dist_{Route})) / Dist_{IRRoute}$;
   return $MIN((1-\beta) \times Cost_{SCRoute}, \beta \times Cost_{IRNN})$;
   }

(그림 4) 공간제약 IRNN질의처리 알고리즘

이러한 공간제약 IRNN질의처리 알고리즘은 도로 내에서 발생할 수 있는 병목현상을 실시간으로 고려하여 좀 더 효율적인 경로설정을 할 수 있는 장점이 있다. 그러나 기존 IRNN질의처리 알고리즘에서 설정된 경로에서 병목현상이 발생된 구간을 제외하고 최적의 새로운 길을 설정해야하는 오버헤드가 발생한다.

4. 결 론 및 향후연구

공간 네트워크 데이터베이스에서는 이동객체가 공간 네트워크상에서만 이동하기 때문에 경로에 기반한 질의의 유용성이 매우 증대되고 있다. 따라서 경로-기반 질의에 대한 효율적인 질의처리 알고리즘 연구가 필수적이다. 대표적인 경로-기반 질의로써 미국Minnesota대학에서 제안한 IRNN질의가 있으며[1], 이는 경로를 가장 적게 벗어나면서 최근접점을 찾는 데에 초점을 맞추고 있다. 그러나 IRNN 질의처리 알고리즘은 첫째, 시간제약을 고려하지 못하기 때문에 시간제약 범위안의 POI(Point of Interest)를 검색하지 못하는 문제점이 발생한다. 둘째, 공간 제약을 고려하지 못하기 때문에 병목현상이 발생한 도로 내의 실시간 상황을 반영하지 못하는 문제점이 존재한다. 이러한 문제점을 극복하기 위해서 본 논문에서는 기존 IRNN 질의처리 알고리즘에서 가장 좋은 성능을 보인 precomputed zone 기반 방법을 기반으로 하여, 시간 및 공간제약을 고려한 IRNN 질의처리 알고리즘을 설계하였다.
향후 연구로는 본 논문에서 설계한 시간 및 공간제약을 고려한 IRNN질의처리 알고리즘을 구현하고, 또한 시간과 공간이 결합된 시공간제약을 고려한 IRNN질의처리 알고리즘 개발하는 것이다.

Acknowledgment

본 논문은 2005년 교육인적자원부의 재원으로 한국학술진흥재단의 지원을 받아 수행된 연구임.
(KRF-2005-041-D00656)

참고문헌

[1]S. Shekhar, and J.S. Yoo, "Processing In-Route Nearest Neighbor Queries: A Comparison of Alternative Approaches", Proc. of ACM GIS, pp9-16, 2003.
[2]P. Sistla, O. Wolfson, S. Chamberlain, and S. Dao, "Modeling and Querying Moving Objects", Proc. of IEEE ICDE, pp422-432, 1997.
[3]Z. Song, and N. Roussopoulos, "K-Nearest neighbor Search for Moving Query Point", Proc. of SSTD, pp79-96, 2001.

[4]Y. Tao, and D. Papadias, "Time Parameterized Queries in Spatio-Temporal Databases", Proc. of ACM SIGMOD, pp334-345, 2002.

[5]Y. Tao, D. Papadias, and Q. Shen, "Continuous Nearest Neighbor Search", Proc. of VLDB, pp287-298, 2002.

[6]M.R. Kolahdouzan, and C. Shahabi, "Continuous K Nearest Neighbor Queries in Spatial Network Databases", Proc. of STDBM, pp33-40, 2004.