분산 그리드 기법을 위한 연속 k-최근접 질의처리 알고리즘

(Countinuous k-Nearest Neighbor Query Processing Algorithm for Distributed Grid Scheme)

김 영 창* 장 재 우**

(Young-Chang Kim) (Jae-Woo Chang)

요 약 최근 GPS 및 무선 이동 컴퓨팅 기술의 발달로 인해, 텔레매틱스(telematics) 및 위치기반 서비스(LBS) 응용이 활발하게 연구되고 있다. 이러한 위치 기반 서비스 응용에서는 이동객체의 위치 정보가 시간의 흐름에 따라 계속적으로 변하기 때문에, 이를 위한 빈번한 업데이트 연산은 시스템에 많은 부하를 가중시키며 이로 인해 검색 성능의 저하를 초래한다. 이를 해결하기 위해 공간 네트워크에서 대용량 이동객체의 위치정보를 분산 처리하기 위한 DS-GRID(distributed S-GRID) 및 이를 위한 k-최근접 질의처리 알고리즘이 제안되었다[1]. 그러나 k-최근접 질의처리 기법은 질의점 및 이동객체의 위치가 변경되면 그 결과가 유효하지 않기 때문에, 연속 k-최근접(CKNN:continuous k-nearest neighbor) 질의처리 알고리즘의 연구가 필요하다. 본 연구에서는 DS-GRID를 위한 MCE-CKNN 알고리즘 및 MBP-CKNN 알고리즘을 제안한다. MCE-CKNN 알고리즘은 주어진 경로를 셀 단위로 분할하여 각 셀에서 질의 처리를 병렬적으로 수행하여 검색 성능을 향상시킨다. 아울러 MBP-CKNN 알고리즘은 그리드 셀의 각 경계점에서 가까운 POI를 미리 저장하여 인접셀 탐색 횟수를 줄임으로써 검색 성능을 향상시킨다. 마지막으로, 제안하는 알고리즘의 성능 분석을 통해, 기존 알고리즘보다 15-53% 검색 성능이 우수함을 나타내었다.

키워드 : 연속 k-최근접 질의, 공간 네트워크, 분산 그리드 기법

Abstract Recently, due to the advanced technologies of mobile devices and wireless communication, there are many studies on telematics and LBS(location-based service) applications. because moving objects usually move on spatial networks, their locations are updated frequently, leading to the degradation of retrieval performance. To manage the frequent updates of moving objects' locations in an efficient way, a new distributed grid scheme, called DS-GRID (distributed S-GRID), and k-NN(k-nearest neighbor) query processing algorithm was proposed[1]. However, the result of k-NN query processing technique may be invalidated as the location of query and moving objects are changed. Therefore, it is necessary to study on continuous k-NN query processing algorithm. In this paper, we propose both MCE-CKNN and MBP(Monitoring in Border Point)-CKNN algorithmss are S-GRID. The MCE-CKNN algorithm splits a query route into sub-routes based on cell and seproves retrieval performance by processing query in parallel way by. In addition, the MBP-CKNN algorithm stores POIs from the border points of each grid cells and seproves retrieval performance by decreasing the number of accesses to the adjacent cells. Finally, it is shown from the performance analysis that our CKNN algorithms achieves 15-53% better retrieval performance than the Kolahdouzan's algorithm.

Keywords : Continuous k-nearest Neighbor Query, Spatial Network, Distributed Grid Scheme

1. 서 론

최근 GPS 및 무선 이동 컴퓨팅 기술의 발달로 인해, 텔레매틱스(telematics) 및 위치기반 서비스(LBS) 응용이 활발하게 연구되고 있다. 위치 기반 서비스의 대상이 되는 객체는 건물과 도로와 같은 정적 위치정보를 갖는 공

† 본 논문은 교육과학기술부와 한국산업기술재단의 지역혁신 인력양성사업으로 수행된 연구결과임.
† 이 논문은 2009년도 정부(교육과학기술부)의 재원으로 한국과학재단의 지원을 받아 수행된 연구임(No. 2009-0059417).

* 전북대학교 컴퓨터공학과 Post. Doc, yckim@dblab.chonbuk.ac.kr
** 전북대학교 컴퓨터공학과 교수, jwchang@chonbuk.ac.kr(교신저자)

논문접수 : 2009.05.26 수정일 : 1차 2009.07.27 심사완료 : 2009.07.29

간객체뿐만 아니라 자동차나 사람과 같은 이동객체를 포함한다. 이동객체는 이상적인 유클리디언(Euclidean) 공간 대신 실제 도로나 철도와 같은 공간 네트워크(sptial network)를 이동하기 때문에, 질의를 처리할 때, POI (points of interests) 및 이동객체 사이의 실제 네트워크 거리를 계산하여 수행한다.

공간 네트워크 상의 이동객체를 위한 기존 질의처리 알고리즘은 네트워크 거리 계산 비용을 감소시키기 위해, 검색대상이 되는 POI와 노드사이의 거리를 미리 계산하여 저장하는 pre-computation 기법을 이용한다[2-5]. 그러나 [2], [3], [4]의 연구는 POI의 위치정보가 변경될 경우 POI와 노드사이의 거리를 재계산해야 하는 단점을 지니며, [5]의 연구는 시간의 흐름에 따라 연속적으로 변경되는 이동객체의 대용량 위치정보를 효율적으로 관리할 수 없다는 단점을 지닌다. 이를 해결하기 위해, 대용량의 이동객체의 위치정보의 관리를 위해 DS-GRID(distributed S-GRID)가 제안되었다[1]. DS-GRID는 공간 네트워크를 균일한 크기의 2차원 그리드 셀 구조로 표현하고 각 그리드 셀을 하나의 서버에 할당하여 이동객체 위치 데이터 관리 및 k-최근접 질의처리를 수행한다.

한편, k-최근접 질의처리 기법은 시간의 흐름에 따라 질의점 및 이동객체의 위치가 변경되면 검색결과가 유효하지 않게 되므로, 이동객체에 대한 연속적인 질의처리 기법인 연속 k-최근접 질의처리 알고리즘이 제안되었다[6]. Kolahdouzan[6]의 연구에서는 유클리디언 공간에서의 연속 k-최근접 질의 연구[7]를 확장하여, 공간 네트워크에서 이동객체를 위한 연속 k-최근접 질의처리 알고리즘을 제안하였다. 그러나 Kolahdouzan의 연구는 NVD (Network Voronoi Diagram)[2]를 이용하여 연속 k-최근접 질의를 수행하기 때문에, POI의 위치정보 변경이 발생할 경우 네트워크의 노드와 노드, 노드와 POI 사이의 네트워크 거리의 재계산으로 인해 검색 성능이 저하된다. 아울러, 연속적으로 위치 정보가 변경되는 이동객체에 대해서는 NVD를 적용하기 어렵다는 단점이 존재한다. 한편, Kolahdouzan의 연구는 단일 시스템을 위한 알고리즘 연구이기 때문에, 실제 응용에서 빈번히 발생하는 대용량 이동객체의 질의 요청을 효율적으로 처리할 수 없다. 즉, 이동객체의 수가 증가할수록 빈번히 발생하는 위치정보 갱신으로 인해 검색 성능이 저하되는 단점을 지닌다. 이를 해결하기 위해, 본 논문에서는, 공간 네트워크 데이터베이스를 위해 제안된 분산 그리드 연구인 DS-GRID[1]를 위한 새로운 연속 k-최근접 질의처리 알고리즘을 제안한다.

본 논문은 구성은 다음과 같다. 2장 관련 연구에서는 공간 네트워크에서 기존 k-최근접 질의처리 및 연속 k-최근접 질의처리 연구를 소개한다. 3장에서는 DS-GRID를 위한 연속 k-최근접 질의처리 알고리즘을 설계하고, 4장에서는 이를 위한 분석적 성능 모델을 제시한다. 5장에서는 제안하는 연속 k-최근접 질의처리 알고리즘의 성능 평가를 수행한다. 마지막으로, 6장에서는 본 논문의 결론 및 향후 연구 방향에 대해서 기술한다.

2. 관련 연구

본 장에서는 공간 네트워크에서 기존 k-최근접 질의처리 알고리즘 연구 및 연속 k-최근접 질의처리 알고리즘 연구를 소개한다.

먼저, 공간 네트워크에서 k-최근접 질의처리 알고리즘 연구인 VN3[2], Island[3], MAK[4], S-GRID[5] 및 DS-GRID[1]의 연구를 소개한다. 첫째, VN3[2] 기법은 유클리디언 환경에서 연구되어진 보로노이 다이어그램을 공간 네트워크에 적용하여 보로노이 영역(Voronoi Polygon)을 생성하고, 해당 영역에 존재하는 각 노드와 노드, POI와 노드 사이의 거리를 미리 계산하여 저장하는 POI 기반 pre-computation 기법을 이용하였다. 이를 통해 k-최근접 질의를 처리시, 질의점이 위치한 보로노이 영역에서 최근접 POI를 검색하고, 인접한 보로노이 영역을 방문하며 k 개의 최근접 POI를 검색한다. 그러나, VN3 기법은 POI의 밀도가 낮거나, k 값이 증가할수록 검색 성능이 저하된다. 또한 POI 및 네트워크 정보의 업데이트가 발생할 경우, 보로노이 영역을 재계산하기 때문에 업데이트에 비효율적인 단점이 존재한다.

둘째, island[3] 기법은 VN3와 마찬가지로 POI 기반 pre-computation 기법을 이용하였다. island 기법은 공간 네트워크의 각 노드로부터 일정 범위 안에 존재하는 POI까지의 거리를 미리 계산하여 island를 생성하여 저장한다. 이를 이용하여, k-최근접 질의 처리시, 질의점이 존재하는 island의 POI를 최근접 POI로 검색하고. 네트워크 확장을 통해 질의점으로부터 가까운 노드를 방문하면서 해당 노드가 속한 island의 POI를 검색한다. island 기법은 island를 형성하기 위한 범위를 조정함으로써 미리 계산하여 저장하기 위한 양을 조절할 수 있기 때문에 VN3 기법에 비해 POI 및 네트워크 데이터의 업데이트에 보다 유연한 구조를 제공한다. 그러나 여전히 POI 및 네트워크 정보의 업데이트 발생시 island 를 재생성해야 하는 단점이 존재한다.

셋째, MAK(Materialization based K-NN)[4]는 VN3 기법의 단점을 해결하기 위해, 노드 기반 pre-computation 기법을 이용하였다. MAK 기법은 질의처리를 수행하기 전에 공간 네트워크의 모든 노드 사이의 거리를 미리 계산하여 저장함으로써 POI의 밀도나 업데이트에 따른 영향 및 질의응답 시간을 최소화하였다. 그러나 공간 네트워크의 모든 노드사이의 거리를 미리 계산하여 저장하기 때문에, 저장공간 오버헤드가 매우 크다. 또한, 노드정보를 업데이트할 경우, 모든 노드사이의 거리를 재계산해야 하는 단점이 존재한다.

넷째, S-GRID[5]는 POI 기반 pre-computation 기법인 VN3, island 기법의 단점을 해결하기 위해, MAK 기법과 마찬가지로 POI 업데이트에 보다 유연한 노드 기반 pre-computation 기법을 이용하였다. S-GRID 기법은 공간 네트워크를 2차원의 고정 크기의 그리드 셀로 나누고, 셀별로 네트워크 정보를 나누어 저장한다. 각 그리드 셀은 셀 영역 내에 포함되는 공간 네트워크의 인접 리스

트 정보 및 각 노드 사이의 거리를 미리 계산하여 Vertex-Edge, Vertex-Border, Cell- Border 의 3개 컴포넌트에 저장한다. S-GRID는 질의점이 존재하는 셀에서의 내부 확장과 인접 셀에서의 외부 확장을 통해 k 최근접 질의 처리를 수행한다. 그러나 S-GRID는 단일 시스템을 고려한 구조이기 때문에, 대용량의 이동객체를 효율적으로 관리하지 못하는 단점이 존재한다.

마지막으로, DS-GRID[1]는 대용량 이동객체의 위치정보를 효율적으로 관리하기 위해, 노드 기반 분산 pre-computation 기법을 이용하였다. 이를 위해, DS-GRID는 공간 네트워크를 2차원의 그리드 셀로 나누고, 네트워크 정보 및 POI와 이동 객체 정보를 셀별로 분산 처리한다. 각 그리드 셀은 셀 테이블(Cell Table), 경계점 테이블(BP Table), POI R-트리, MO Index, Vertex-Edge, Vertex-Border, Cell-Border의 6개 컴포넌트로 구성되며, 서버마다 하나의 그리드 셀이 할당되어 각 셀 영역에 포함되는 공간 네트워크 정보를 독립적으로 관리한다. 아울러 DS-GRID[1]에서는 k-최근접 질의처리를 위해 ICE (incremental cell expansion) 알고리즘과 MCE(multi-castring-based cell expansion) 알고리즘을 제안하였다. ICE 알고리즘은 질의점이 위치한 셀과 인접 셀이 공유하는 모든 경계점 정보를 한 번의 질의 전송을 통해 전달함으로써, 인접 셀에서의 POI 재탐색 횟수를 감소시켜 검색 성능을 향상시켰다. 아울러, MCE 알고리즘은 k 개의 POI를 검색하기 위한 셀 리스트를 생성하여, 이웃 셀에서의 POI 탐색을 병렬적으로 수행하여 검색 성능을 향상시켰다. 그러나 k-최근접 질의처리 알고리즘은 정적인 위치로부터 가까운 POI를 검색하기 때문에, 시간의 흐름에 따라 질의점 및 이동객체의 위치가 변경되면 검색결과가 유효하지 않게 된다. 이를 해결하기 위해서는 이동객체에 대한 연속적인 질의처리 기법인 연속 k-최근접 질의처리 알고리즘의 연구가 필요하다.

공간 네트워크에서 이동객체를 위한 연속 k-최근접 질의 처리 알고리즘 연구로는 Kolahdouzan[6]의 연구가 유일하다. Kolahdouzan[6]의 연구에서는 NVD(Network Voronoi Diagram)[2]를 이용하여 연속 k-최근접 질의처리를 위한 IE(Intersection Examination)과 UBA (Upper Bound Algorithm)의 두 알고리즘을 제안하였다. 첫째, IE 알고리즘은 이동 객체가 이동하고자 하는 경로상의 모든 노드에서 k-최근접 질의 처리 알고리즘을 수행하여 k 개의 POI를 검색한다. 아울러 각 노드에서 검색된 POI를 increasing 거리와 decreasing 거리를 갖는 POI로 구분하고 연속된 increasing POI와 decreasing POI의 쌍의 최소 거리를 이용하여 분할점을 계산한다. 아울러 이러한 과정을 모든 부분 경로에 대해 수행하여 결과집합을 생성한다. 둘째, UBA 알고리즘은 IE 알고리즘이 경로상의 노드에서 k 최근접 질의 처리 알고리즘을 수행해야하는 오버헤드를 줄이고자 Roussopoulos [7]의 연구에서 제안된 임계값(threshold) δ값을 이용하였다. 먼저 시작 노드에서 k+1개의 POI를 검색한 후 이를 increasing 거리와 decreasing 거리를 갖는 POI로 구분한다. IE 알고리즘과 마찬가지로 연속된 increasing POI와 decreasing POI 쌍의 최소 거리를 구하여 δ값을 계산한다. 이때 δ값 범위 내에서는 검색된 POI의 거리 순서가 변하지 않는다는 이론을 증명하고, 이를 이용하여 k-최근접 질의 처리 알고리즘의 수행 횟수를 감소시켜 검색시간을 향상시켰다. 그러나 Kolahdouzan의 연구는 NVD를 이용하여 질의처리를 수행하기 때문에, POI 및 네트워크의 업데이트에 취약하며, 대용량 이동 객체를 효과적으로 지원하지 못하는 단점을 갖는다.

3. 분산 연속 k-최근접 질의처리 알고리즘의 설계

대용량 이동 객체를 위한 분산 처리 연구인 DS-GRID를 이용하여 새로운 연속 k-최근접 질의처리 알고리즘인 MCE-CKNN 알고리즘과 MBP-CKNN 알고리즘을 제안한다.

3.1 MCE-CKNN(MCE-based continuous k-Nearest Neighbor) 알고리즘

연속 k-최근접 질의를 처리하기 위해, 이동경로 상의 모든 지점에서 연속적인 k-최근접 질의 처리 알고리즘 수행은 많은 검색시간을 요구한다. 이때 공간 네트워크에서의 이동경로는 교차로와 같은 노드와 도로와 같이 두 개의 노드로 구성되는 에지의 집합으로 표현할 수 있다. 이동 객체가 에지 상에서 이동할 때 에지상의 임의의 지점에서 가장 가까운 k 개의 POI는 에지의 양 노드에서 검색한 k 개의 POI중에 포함되기 때문에 k-최근접 질의처리 알고리즘의 수행 횟수를 감소시킬 수 있다. 즉, 공간 네트워크에서는 이동 객체가 이동하고자 하는 이동경로상의 노드의 수만큼의 k-최근접 질의 처리 알고리즘 수행으로 결과 집합을 생성할 수 있다. 그러나 이동경로상의 노드의 수만큼의 k-최근접 질의 처리 알고리즘 수행이 필요하다. 이를 해결하기 위해, DS-GRID를 위한 연속 k-최근접 질의처리 알고리즘인 MCE-CKNN 알고리즘을 제안한다.

MCE-CKNN 알고리즘은 k-최근접 질의처리 알고리즘의 수행 횟수를 감소시키기 위해, Kolahdouzan[6]의 연구에서 정의한 임계값(threshold) δ를 이용하고자 한다. 이에 따른 MCE-CKNN 알고리즘의 수행 절차는 다음과 같다. 첫째, 이동 객체의 이동경로가 지나는 셀의 리스트 CellList={$Cell_1$, $Cell_2$, ..., $Cell_n$} 를 생성한 후 이동경로상에 존재하는 노드정보를 노드가 포함된 셀에 삽입한다. 이때 셀 정보는 $Cell_i$={N_1, N_2,...N_m} 로 구성되며 $Cell_i$는 셀 리스트에 포함된 i 번째 셀을, N 은 해당 셀에 포함되는 노드를 나타내며, m은 이동경로 상에 존재하는 노드 중 i 번째 셀에 포함되는 노드의 수를 나타낸다. 둘째, 생성된 셀 정보를 해당 셀을 담당하는 서버로 전송하여 각 서버에서 병렬적으로 연속 k-최근접 질의 처리 알고리즘을 수행한다. 셋째, 셀 정보를 전달받은 서버에서는 각 셀에서 연속 k-최근접 질의 처리 알고리즘을 수행한다.

이때 각 셀에서의 연속 k-최근접 질의 처리는 MCE 알고리즘과 [1]에서 제안한 δ값을 이용하여 수행하며, 수행 절차는 다음과 같다. 첫째, 전달받은 셀 정보에 포함된 첫 번째 노드에서 MCE 알고리즘을 이용하여 가장 가까운 k+1 개의 POI를 검색한다. 둘째, 검색된 k+1 개의 POI를 거리 순으로 정렬하여 [6]에서 정의한 increasing 거리와 decreasing 거리를 갖는 POI로 분류한 후 δ값을 계산한다. 정렬된 POI 집합을 {P_1↑, P_2↓, P_3↑, ...,P_{k+1}↓}라 가정할 때, ↑는 increasing 거리를 ↓는 decreasing 거리를 나타내며 δ값은 다음과 같다.

$$\delta = \min(distance(P_i,q) - distance(P_{i+1},q)) \quad (1)$$

이때 q는 질의점을, P_i, P_{i+1}은 각각 increasing 거리와 decreasing 거리를 갖는 POI를 나타내며, distance(P, q)는 P와 q의 최단 거리를 나타낸다. 셋째, 계산된 δ값보다 큰 거리를 갖는 다음 노드에서 k+1 개의 POI 검색과 새로운 δ값의 계산을 수행하고, 이를 셀 정보에 포함된 마지막 노드까지 반복 수행한다. 마지막으로, 검색된 결과를 전송하여 병합한 후 사용자에게 결과를 반환한다.

그림 1은 제안하는 MCE-CKNN 알고리즘을 나타낸다. 제안하는 알고리즘은 크게 질의로 주어진 이동경로를 이용하여 셀 리스트를 생성하여 각 셀 담당 서버에 전달하고 결과를 병합하는 MCE-CKNN Master 및 각 셀에서 전달받은 셀 정보를 이용하여 실제 연속 k-최근접 질의 처리 알고리즘을 수행하는 MCE-CKNN Worker 로 구성된다.

MCE-CKNN Master 알고리즘은 첫째, 질의로 주어진 이동경로(NodeList) 에 포함된 노드 정보를 이용하여 셀 리스트를 생성하고 노드 정보를 해당 셀에 삽입한다(1-2 라인). 둘째, 생성된 셀 정보를 각 셀 담당 서버에 전달하여 MCE-CKNN Worker 알고리즘을 수행한다(3-4 라인). 마지막으로, 각 셀에서 검색된 결과를 전달받아 이를 병합한 후 사용자에게 결과를 반환한다(5-7 라인).

각 셀에서 수행되는 MCE-CKNN Worker 알고리즘은 첫째, 전달받은 셀 정보에 포함된 첫 번째 노드에서 MCE 알고리즘을 이용하여 k+1 개의 POI 를 검색한 후, 검색된 POI를 increasing 거리와 decreasing 거리를 갖는 POI로 구분한다(8-9 라인). 둘째, 검색된 k+1개의 POI를 이용하여 δ값을 계산한다(10-11 라인). 셋째, 계산된 δ값보다 큰 거리를 갖는 다음 노드를 검색한 후, 검색된 노드에서 MCE 알고리즘을 이용하여 k+1개의 POI를 검색한다(12-13). 넷째, POI를 분류하고 노드 N_i와 N_q 사이의 분할점(split point)를 구한다(14-15). 다섯째, N_q와 다음 노드 사이의 연속 k-최근접 POI를 검색하기 위해 N_q를 시작 노드로 설정하고, 검색된 POI값을 δ값을 구하기 위한 후보 집합으로 설정한다. 마지막으로 이를 셀 정보에 포함된 모든 노드에 대해 반복한 후 결과 값을 반환한다.

MCE-CKNN Master(NodeList, k)

CellList=∅, result=∅
01. for each N_i∈NodeList
02. CellList.update(findCell(N_i), N_i)
03. for each $Cell_i$∈CellList
04. sendQuery($Cell_i$, k)
05. for each $Cell_i$∈CellList
06. result.update(ReceiveResult($Cell_i$))
07. return result

MCE-CKNN worker(Cell, k)

result=∅, $candidate_i$=$candidate_{i+1}$=∅
08. MCE(N_0, k+1, $candidate_i$)
09. computeDirection($candidate_i$)
10. for each N_i∈Cell start from N_1
11. δ=computeDelta($candidate_i$)
12. N_q=findNextN(Cell, δ)
13. MCE(N_q, k+1, $candidate_{i+1}$)
14. computeDirection($candidate_{i+1}$)
15. result+= findSplitPoints($candidate_i$, $candidate_{i+1}$)
16. N_i=N_q
17. $candidate_i$=$candidate_{i+1}$
18. return result

그림 1. MCE-CKNN 알고리즘

MCE-CKNN 알고리즘은 각 셀에서의 연속 k-최근접 질의 처리 시, 이동경로 상에 존재하는 노드에서 MCE 알고리즘을 이용하여 k+1개의 POI를 검색한다. 그림 2는 MCE-CKNN 알고리즘의 예를 나타낸다.

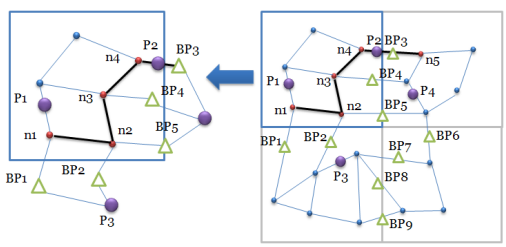

그림 2. MCE-CKNN 알고리즘의 예

이동 객체가 {n_1, n_2, n_3, n_4, n_5} 로 구성되는 이동경로를 운행할 때, 연속 2-최근접 질의 처리 알고리즘을 가정한다. 먼저, {n_1, n_2, n_3, n_4}는 같은 셀에 존재하므로 노드 n_1에서 MCE 알고리즘을 이용하여 2+1 개의 POI P_1, P_2, P_3 를 검색한다. 이때, POI P_1과 P_2는 내부 확장을 통해서 검색되며, P_3는 질의점이 위치한 셀의 경계점 BP_1, BP_2 를 공유하는 인접 셀에서 외부 확장을 이용하여 검색한다. 아울러, 계산된 δ값이 에지 (n_1, n_2)의 길이보다

작을 경우, 노드 2에서도 마찬가지 방법으로 2+1 개의 POI P_1, P_2, P_3를 검색하며, P_1, P_2는 내부 확장을 통해, P_3는 경계점 BP_1, BP_2를 공유하는 인접 셀에서 외부 확장을 이용하여 검색한다. 노드 n_3, n_4에서는 2+1개의 POI P_1, P_2, P_4를 검색하며, P_1, P_2는 내부 확장을 통해, P_3는 경계점 BP_3, BP_4, BP_5를 이용하여 외부 확장을 이용하여 검색한다. 즉, 노드 n_1, n_2, n_3, n_4의 2+1 NN을 검색하기 위하여 최악의 경우 인접 셀에서 노드 개수인 4회의 외부 확장을 수행하는 경우가 발생한다. 이러한 현상은 이동 객체의 이동경로가 길어질수록, 그리고 셀의 크기가 증가할수록 셀 안에 포함되는 노드의 수가 증가되어, 유사한 검색을 반복적으로 수행함으로써 전체적인 질의 처리의 효율성을 저하시킨다.

3.2 MBP(Monitoring in Border Point)-CKNN 알고리즘

MCE-CKNN 알고리즘의 단점을 해결하기 위해 경계점에서 가장 가까운 k 개의 POI를 미리 계산하여 저장하고, 이를 이용하여 질의 처리의 효율성을 높이기 위한 MBP-CKNN 알고리즘을 제안한다. MBP-CKNN 알고리즘을 처리하기 위해서는 각 셀에 존재하는 모든 경계점에 대하여 가장 가까운 k 개의 POI를 미리 계산하고 이를 저장해야 한다. 이는 POI가 위치 정보의 갱신이 빈번하지 않은 정적인 데이터이기 때문에, DS-GRID의 네트워크 데이터와 함께 생성하여 사용할 수 있다. 즉, DS-GRID를 위한 네트워크 데이터를 생성한 후, POI 데이터 삽입 시 Vertex-Border 컴포넌트를 이용하여 셀 내에 존재하는 모든 경계점까지의 거리를 계산하고 각 경계점으로부터 가장 가까운 k 개의 POI를 저장한다.

그림 3은 MBP를 위한 k를 2라 가정할 때의 MBP-CKNN 알고리즘의 예를 나타낸다. 이동 객체가 {n_1, n_2, n_3, n_4, n_5} 로 구성되는 이동경로를 운행할 때 연속 2-최근접 질의 처리 알고리즘을 가정한다. 이때 MBP 파일을 생성하기 위한 POI의 수는 1이라 가정한다. 노드 {n_1, n_2, n_3, n_4}는 같은 셀에 존재하므로 첫 번째 노드 n_1에서 MCE 알고리즘을 이용하여 내부 확장을 수행하여 2개의 POI P_1, P_2 를 검색한다. 이때 2+1=3 번째의 POI를 검색하기 위해 각 경계점 BP_1, BP_2, BP_3, BP_4, BP_5를 공유하는 인접 셀 담당 서버로 각 경계점에서 가장 가까운 1개의 POI 정보를 요청한다. 셀 1(cell 1)의 경우 BP_3, BP_4, BP_5에서 가장 가까운 POI는 P_4, 셀 2(cell 2)의 경우 BP_1, BP_2에 가장 가까운 POI는 P_3 이므로 셀 0은 이를 전달받아 MBP 리스트를 구축한다.

이를 이용하여 첫 번째 노드 n_1 으로부터 가까운 3 번째의 POI를 검색한다. 노드 n_2 에서도 마찬가지로 내부 확장을 통해 2개의 POI P_1, P_2를 검색하며 3번째의 POI를 검색한다. 이때 구축된 MBP List를 이용하여 추가적인 외부 확장 없이 3번째의 POI를 검색할 수 있다.

MBP-CKNN 알고리즘은 MBP-CKNN Master 알고리즘과 MBP- CKNN Worker 알고리즘으로 구성되며, MBP-CKNN Master 알고리즘은 MCE-CKNN Master

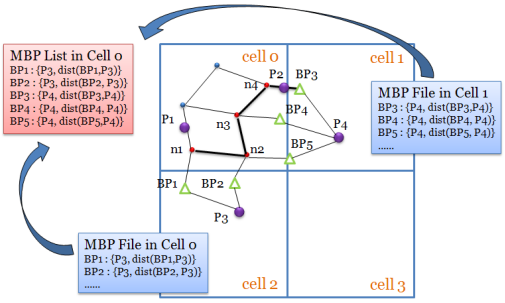

그림 3. MBP-CKNN 알고리즘의 예

알고리즘과 동일하다. 그림 4는 각 셀에서 실행되는 MBP-CKNN Worker 알고리즘을 나타낸다. 첫째, 전달받은 셀 정보에 포함된 첫 번째 노드에서 내부 확장(InnerExpantion)을 통해 k+1 개의 POI 를 검색한 후, 셀의 각 경계점을 공유하는 셀 리스트를 생성한다(1-3 라인). 둘째, 셀 리스트에 존재하는 각 셀로 질의를 전달하여 이웃 셀의 MBP 파일에 저장된 POI 정보를 전달받아 MBPList 를 생성한다(4-7 라인). 셋째, 내부 확장을 통해 검색된 k+1번째의 POI보다 노드 N0까지의 거리가 작은 모든 경계점에 대해 MBPList에 저장된 POI정보를 이용하여 candidatei를 갱신한다(8-13 라인). 넷째, k+1개의 POI를 를 increasing 거리와 decreasing 거리를 갖는 POI로 구분하고, 검색된 k+1개의 POI를 이용하여 δ값을 계산한다(14-16 라인). 다섯째, 계산된 δ값보다 큰 거리를 갖는 다음 노드를 검색한 후, 검색된 노드에서 내부 확장과 MBPList의 POI 정보를 이용하여 k+1개의 POI를 검색한다(17-24 라인). 여섯째, 마찬가지 방법으로 POI를 분류하고 노드 N_i와 N_q 사이의 분할점(split point)를 구한다(25-26 라인). 다섯째, N_q와 다음 노드 사이의 연속 k-최근접 POI를 검색하기 위해 N_q를 시작 노드로 설정하고 검색된 POI값을 δ값을 구하기 위한 후보 집합으로 설정한다(27-28 라인). 마지막으로 이를 셀 정보에 포함된 모든 노드에 대해 반복한 후 결과 값을 반환한다(29 라인).

MBP-CKNN worker(Cell, k)

CellList=∅, MBPList=∅, result=candidate$_0$=candidate$_{i+1}$=∅

01. **InnerExpantion**(N_0, k+1, candidate$_i$, BPList)
02. for each BP_i∈BPList
03. CellList.update(findCell(BP_i), BP_i)
04. for each Cell$_i$∈CellList
05. sendQuery(Cell$_i$, k)
06. for each Cell$_i$∈CellList
07. MBPList.update(ReceiveResult(Cell$_i$))
08. dmax=candidate$_i$.k+1thDist()
09. for any BP_i∈MBPList && dist(N_0, BP_i)<dmax)
10. for each POI_i∈BP_i.POI
11. if((dist(N_0, BP_i)+POI_i.dist)<dmax)

```
12.          candidate_i.update(POI_i,
                              (dist(N_0, BP_i)+POI_i.dist))
13.          dmax=candidate_i.k+1thDist()
14. for computeDirection(candidate_i)
15. for each N_i∈Cell start from N_1
16.     δ=computeDelta(candidate_i)
17.     N_q=findNextN(Cell, δ)
18.     InnerExpantion(N_q, k+1, candidate_i+1, BPList)
19.     dmax=candidate_i+1.k+1thDist()
20.     for any BP_i∈MBPList
                              && dist(N_0, BP_i)<dmax)
21.         for each POI_i∈BP_i.POI
22.             if((dist(N_0, BP_i)+POI_i.dist)<dmax)
23.                 candidate_i+1.update(POI_i, (dist(N_0,
                                      BP_i)+POI_i.dist))
24.                 dmax=candidate_i+1.k+1thDist()
25.     computeDirection(candidate_i+1)
26.     result+= findSplitPoints(candidate_i, candidate_i+1)
27.     N_i=N_q
28.     candidate_i=candidate_i+1
29. return result
```

그림 4. MBP-CKNN worker 알고리즘

4. 분석적 성능 모델

제안하는 분산 연속 k-최근접 질의 처리 알고리즘의 성능 비교를 위해 분석적 성능 모델을 제시한다. 이를 위해 노드 및 에지, POI의 밀도가 일정한 정방형 공간 네트워크를 가정한다.

4.1 Kolahdouzan[6]의 연속 k-최근접 질의 처리 알고리즘을 위한 분석적 성능 모델

제안하는 연속 k-최근접 질의 처리 알고리즘과의 분석적 성능 평가를 수행하기 위해 기존 연구인 Kolahdouzan[6]의 연속 k-최근접 질의 처리 알고리즘을 위한 분석적 성능 모델을 제시한다. Kolahdouzan의 연속 k-최근접 질의 처리 알고리즘은 주어진 경로의 시작 노드에서 k+1 개의 POI를 검색하여 increasing, decreasing 거리를 갖는 POI로 구분한 후, 임계값 δ를 계산하고 분할점을 생성한다. 아울러 이를 경로의 마지막 노드까지 반복 수행하여 결과 집합을 생성한다. 따라서 Kolahdouzan의 연속 k-최근접 질의 처리 알고리즘 비용은 크게 각 노드에서의 k+1 개의 POI를 검색하기 위한 k-최근접 질의 처리 알고리즘 수행 비용과 분할점 생성 비용의 합으로 계산할 수 있다.

첫째, Kolahdouzan의 알고리즘은 각 노드에서 k 개의 POI를 검색하기 위해 NVD를 이용한다. NVD는 POI를 중심으로 NVD 영역에 포함되는 POI와 노드 및 경계점 사이의 거리를 미리 계산하여 저장한다. 따라서 노드에서의 k-최근접 질의 처리 비용은 NVD에 대한 디스크 I/O를 구하여 측정할 수 있다. NVD는 크게 다음과 같은 4개의 정보로 구성된다.

- NVD : 보로노이 폴리곤(polygon) 영역 및 POI 정보
- AVNP : 인접한 다른 POI의 NVD 정보
- BNVP : 보로노이 폴리곤의 각 노드 및 경계점 사이의 연결 정보
- PCC : 보로노이 폴리곤의 POI, 노드, 경계점 사이의 거리 정보

k-최근접 질의 처리 알고리즘은, 첫째, 질의점이 위치한 보로노이 폴리곤 영역의 정보를 검색한다. 이는 NVD 정보를 이용하여 검색하며 NVD는 {폴리곤ID, MBR, POIID, POI 좌표} 정보로 구성된다. 둘째, ANVP 정보를 이용하여 인접 폴리곤을 검색하고 BNVP, PCC 정보를 이용하여 인접 폴리곤의 POI 까지의 거리를 계산한다. 마지막으로 검색된 인접 폴리곤의 경계점보다 적은 거리를 갖는 k 개의 POI 를 검색할 때까지 반복한다. 따라서 k 최근접 질의 처리 비용은 4개의 정보에 대한 디스크 I/O 횟수의 합으로 계산할 수 있다. 각 폴리곤은 NVD, AVNP 정보 검색을 위한 두 번의 디스크 접근과, 질의점이 위치한 에지의 두 노드와 폴리곤 내의 모든 경계점까지의 거리를 검색하기 위한 경계점의 수*2 번의 디스크 접근이 이루어진다. 따라서 폴리곤의 디스크 접근 비용은 다음과 같다.

$$COST_{NVD} = 2 + \left\lceil \frac{recordsize_{BNVP} * \#ofBP_i * 2}{pagesize} \right\rceil + \left\lceil \frac{recordsize_{PCC} * \#ofBP_i * 2}{pagesize} \right\rceil \quad (2)$$

아울러 k-최근접 질의 처리 알고리즘은 k 개의 POI를 검색할 때까지 k 번째 POI 보다 작은 거리를 갖는 경계점의 이웃 폴리곤을 검색한다. 따라서 k-최근접 질의 처리 알고리즘의 검색 비용은 다음과 같다.

$$COST_{NVD-KNN} = COST_{NVD} * (k+1) + COST_{NVD} * \#ofadjNVP_i \quad (3)$$

마지막으로 연속 k-최근접 질의 처리 알고리즘은 주어진 경로의 노드에서 k-최근접 질의 처리 알고리즘을 수행하기 때문에, 주어진 경로의 노드의 수가 n 일때의 연속 k-최근접 질의 처리 알고리즘 비용은 다음과 같다.

$$COST_{UBA} = COST_{NVD-KNN} * n$$

4.2 MCE-CKNN 알고리즘의 분석적 성능 모델

MCE-CKNN 알고리즘의 분석적 성능 모델을 위해, 이동 객체가 N개의 노드로 구성된 경로를 이동한다고 가정한다. MCE-CKNN 알고리즘은, 첫째, 이동 객체가 이동하고자 계획한 경로를 노드의 좌표를 이용하여 셀 단위의 부분 경로로 분할한다. 둘째, 분할된 부분 경로가 속한 셀 담당 서버로 질의를 전송한다. 셋째, 질의를 전송받은 각 서버는 분할된 부분 경로의 모든 노드에서 MCE 알고리즘을 이용하여 k 개의 POI를 검색하고, 검색된 POI를 이용하여 셀 내에 포함된 노드 사이의 분할점을 생성한다. 넷째, 각 셀에서 부분 경로에 대해 검색된 결과를 질의를 전달한 셀로 전송하여 병합한다. 마지

막으로 분할점 계산이 이루어지지 않은 경계점을 포함하는 에지에 대해 분할점을 계산하여 결과 집합을 생성한다. 따라서 MCE-CKNN 알고리즘은 크게 다섯 단계로 구성되며 각각에 대한 비용 정의 및 이에 대한 설명은 다음과 같다.

- $COST_{createsubroute}$: 이동 경로를 셀 단위의 부분 경로로 분할
- $COST_{transmit}$: 각 부분 경로를 포함하는 영역의 셀 담당 서버로 전송
- $COST_{subCKNN}$: 부분 경로에 대한 연속 k-최근접 질의 처리 수행
- $COST_{receive}$: 부분 경로에 대해 검색된 결과 전송
- $COST_{splitsection}$: 경계점을 포함하는 에지에서의 분할점 생성

DS-GRID의 질의 처리 알고리즘은 일반적으로 디스크 I/O 및 분산 저장된 데이터 검색을 위한 네트워크 통신 비용을 구하여 분석적 성능을 측정할 수 있다. 이때, $COST_{createsubroute}$, $COST_{slitsection}$은 주어진 경로와 검색된 부분 결과를 이용하여 메모리 상에서 수행되므로 전체 검색 비용에 미치는 영향이 매우 적다. 따라서 MCE-CKNN 알고리즘 비용은 $COST_{transmit}$, $COST_{subCKNN}$, $COST_{receive}$ 비용의 합으로 계산한다.

첫째, 각 셀로 질의를 전송하기 위한 비용은 질의를 각 서버로 전송하기 위해 필요한 메시지의 수와 하나의 메시지를 전송하기 위한 네트워크 통신 비용의 곱으로 계산할 수 있다. 각 서버로 전송하기 위해 필요한 메시지의 수는 이동 경로가 통과한 셀의 수를 이용하여 계산할 수 있다. 하나의 셀은 x, y축 방향으로 n 개의 노드로 구성되므로, N 개의 노드로 구성된 이동 경로가 통과하는 셀의 수는 궤적의 길이를 셀의 x, y 축 당 노드 개수인 n으로 나누어 계산할 수 있다. 따라서 N 개의 노드로 구성된 이동 경로가 통과하는 셀의 수는 다음과 같다.

$$\#ofCell_{route} = N/n \tag{4}$$

이때, 질의 전송은 부분 경로가 통과하는 셀 중 질의를 처리하는 셀을 제외한 셀에 대해서만 이루어진다. 따라서 질의를 각 셀 담당 서버로 질의를 전송하기 위한 비용은 다음과 같다.

$$COST_{transmit} = Cost_{communication} * \left\lceil \frac{querysize}{packetsize} \right\rceil * (ofCELL_{route} - 1) \tag{5}$$

둘째, 각 셀에서 부분 경로에 대한 연속 k-최근접 질의 처리 수행은 부분 경로를 구성하는 각 노드에서의 k개의 POI를 검색하기 위한 MCE 알고리즘의 비용 및 검색된 k 개의 POI를 이용하여 각 노드 사이의 분할점을 계산하기 위한 두 부분으로 나눌 수 있다. 이때, 분할점 계산은 검색된 POI를 이용하여 메모리상에서 이루어지므로, $COST_{subCKNN}$는 이를 제외한 MCE 알고리즘의 검색 비용으로 측정할 수 있다. 이때 MCE 알고리즘은 각 서버에 할당된 부분 경로를 구성하는 노드 수 n만큼 반복 수행 되기 때문에 부분 경로에 대한 연속 k-최근접 질의 처리를 위한 비용은 다음과 같다.

$$COST_{subCKNN} = COST_{MCE} * n \tag{6}$$

마지막으로, 각 서버에서 검색된 부분 경로에 대한 연속 k-최근접 질의 처리 결과 전송 비용은 질의 전송 비용과 마찬가지로 결과 전송에 필요한 메시지의 수와 하나의 메시지를 전송하기 위해 요구되는 네트워크 통신 비용의 곱으로 계산할 수 있다. 따라서 검색된 부분 경로에 대한 결과 전송 비용은 다음과 같다.

$$COST_{receive} = Cost_{communication} * \left\lceil \frac{k*\#ofSplitSection}{packetsize} \right\rceil \tag{7}$$

마지막으로 MCE-CKNN 알고리즘 수행에 비용은 다음과 같다.

$$COST_{MCE-CKNN} = COST_{trasmit} + COST_{subCKNN} + COST_{receive}$$

4.3 MBP-CKNN 알고리즘의 분석적 성능 모델

MBP-CKNN 알고리즘은 MCE-CKNN 알고리즘과 마찬가지로 다섯 단계로 이루어진다. 따라서 전체 질의 처리 비용은 각 단계의 비용의 합으로 계산할 수 있다. 즉, 첫째, 이동 객체가 이동하고자 계획한 경로를 노드의 좌표를 이용하여 셀 단위의 부분 경로로 분할 비용, 둘째, 분할된 부분 경로를 해당 셀 담당 서버로 질의를 전송하기 위한 통신비용, 셋째, 각 셀 담당 서버에서 분할된 부분 경로에 대한 연속 k-최근접 질의 처리 비용, 넷째, 각 셀에서 부분 경로에 대해 검색된 결과를 전송하기 위한 통신비용, 마지막으로 분할점 계산이 이루어지지 않은 경계점을 포함하는 에지에 대해 분할점을 계산비용이 요구된다. 또한 MCE-CKNN 알고리즘 비용과 마찬가지로 $COST_{createsubroute}$, $COST_{slitsection}$은 전체 검색 비용에 미치는 영향이 매우 적기 때문에, $COST_{transmit}$, $COST_{subCKNN}$, $COST_{receive}$ 비용의 합으로 계산한다.

첫째, MBP-CKNN 알고리즘을 위한 $COST_{transmit}$, $COST_{receive}$ 비용은 식 (5), (7)을 이용하여 동일하게 계산할 수 있다.

둘째, MBP-CKNN 알고리즘은 각 서버에서 부분 경로에 대한 연속 k-최근접 질의 처리를 수행할 때, 셀마다 경계점으로부터 가까운 k 개의 POI를 미리 계산하여 저장하고 이를 이용하여 한 번의 외부 확장만을 수행한다. 따라서 부분 경로에 대한 연속 k-최근접 질의 처리 비용은 부분 경로를 구성하는 각 노드에서의 셀 내부 확장 비용 $COST_{innerexpansion}$ 및 한 번의 외부 확장 비용 $COST_{outerexpansion}$의 합으로 계산할 수 있다. 먼저 $COST_{innerexpansion}$은 3.3.3 절의 S-GRID를 위한 k-최근접 질의 처리 알고리즘의 Vertex-Border 컴포넌트 검색 비용 및 Vertex-Edge 컴포넌트 검색 비용을 이용하여 계산할 수 있다. 이때 내부 확장은 부분 경로를 구성하는 노드의 수만큼 반복된다. 따라서 부분 경로에 대한 연속 k-최근접 질의 처리의 내부 확장비용은 다음과 같다.

$$COST_{innerexpansion} = (COST_{inner}(V-B) + COST_{inner}(V-E))*n \quad (8)$$

MBP-CKNN 알고리즘은 부분 경로를 구성하는 노드 수 만큼의 k-최근접 질의 처리 수행에 따른 외부 확장 비용을 줄이기 위해 각 셀의 경계점에서 가까운 k 개의 POI를 미리 계산하여 저장한다. 따라서 외부 확장비용은 셀에 저장된 각 경계점으로부터 가까운 k 개의 POI 를 검색하기 위한 디스크 I/O 및 이를 위한 질의 및 결과를 전송하기 위한 네트워크 통신비용의 합으로 계산할 수 있다. 먼저 경계점으로부터 가까운 k 개의 POI 검색을 위한 디스크 I/O 는 인접한 셀과 공유하는 경계점의 수, 각 경계점에서의 POI 정보 접근 비용의 곱으로 계산할 수 있다. 노드 및 에지의 밀도가 일정한 정방형 공간 네트워크에서의 인접한 셀의 수는 4이므로, 이웃 셀에서 #ofBP$_i$/4개의 경계점에 대한 k 개의 POI를 검색해야 한다. 따라서 경계점으로부터 가까운 k 개의 POI 검색을 위한 디스크 접근 비용은 다음과 같다.

$$COST_{outerexpansion}(MBP) = \frac{\#ofBP}{4} * COST_{MBP} \quad (9)$$

마지막으로 외부 확장을 위한 질의 전송 비용 및 결과 전송은 식 (5), 식 (7)과 유사하게 계산되며, 이는 다음과 같다.

$$COST_{transmit}(MBP) = Cost_{communication} * \left\lceil \frac{querysize}{packetsize} \right\rceil \quad (10)$$

$$COST_{result}(MBP) = Cost_{communication} * \left\lceil \frac{\#ofBP * k * recordsize_{MBP}}{packetsize} \right\rceil \quad (11)$$

따라서 분할된 부분 경로에 대한 연속 k-최근접 질의 처리 비용은 부분 경로의 각 노드에서의 내부 확장 비용, 경계점에서의 POI 검색을 위한 질의 전송 비용, 이웃 셀에서의 경계점에서 가까운 k 개의 POI 검색 비용, 그리고 검색된 결과의 전송 비용의 합으로 계산할 수 있으며, 이는 다음과 같다.

$$COST_{subCKNN} = COST_{innerexpansion} + COST_{outerexpansion}(MBP) + COST_{transmit}(MBP) + COST_{result}(MBP) \quad (12)$$

마지막으로 MBP-CKNN 알고리즘 수행에 비용은 다음과 같다.

$$COST_{MBP-CKNN} = COST_{trasmit} + COST_{subCKNN} + COST_{receive}$$

5. 성능 평가

본 장에서는 기존 Kolahdouzan[6]의 연속 k-최근접 알고리즘 및 제안하는 MCE-CKNN, MBP-CKNN 알고리즘의 검색 성능을 비교한다. 성능 평가 환경은 HP ML 150 G3 서버, 인텔 Xeon 3.0 Ghz dual CPU, 2 GB 메모리, HP 250 GB SATA 7200 rpm HDD, BCM5703 Gigabyte Ethernet 이며, visual studio 2003을 사용하여 제안하는 알고리즘을 구현하였다. DS-GRID는 멀티 프로세스를 이용하여 구현하였으며, 하나의 프로세스에 하나의 셀을 할당하고, 네트워크 통신을 통해 질의 할당하고를 전달한다. 성능 평가를 위한 공간 네트워크 데이터는 220,000 개의 에지와 170,000 개의 노드로 구성된 샌프란시스코 만 데이터를 이용하였으며, POI와 이동 객체는 Brinkhoff 알고리즘[8]을 이용하여 생성하였다. POI 데이터는 에지 대비 POI 밀도 0.01(2200개을 통0.02(4400개을 통0.05(11000개을 통0.1(22000개을의 데이터 집합을 생성하고, R-트리를 이용하여 색인하였다. 사용된 R-트리의 fan-out은 50이며, 평균 1.4의 깊이를 갖는다. 질의 데이터는 경로의 길이를 10, 20, 30, 40, 50으로 생성하였으며, 경로의 길이별로 100개의 질의 데이터를 생성하였다. 성능 평가를 위해, 첫째, 경로의 길이 변화에 따른 성능 비교, 둘째, k 값의 변화에 따른 성능 비교, 셋째, POI의 밀도에 따른 성능 비교를 수행하고, 마지막으로 제안하는 분석적 성능모델과의 정확성 분석을 수행한다.

첫째, 질의 경로의 길이 변화에 따른 검색 성능을 비교한다. 이때, k 는 20, POI 밀도는 0.01, 그리드 셀의 개수는 10*10 이다. 그림 5는 질의 경로의 길이 변화에 따른 검색 성능을 나타낸다. 제안하는 연속 k-최근접 질의 처리 알고리즘의 성능이 Kolahdouzan의 방법보다 검색성능이 우수함을 알 수 있다. 이는 제안하는 연속 k-최근접 질의 처리 알고리즘은 주어진 경로를 셀 단위의 부분 경로로 분할하여 각 셀 담당 서버로 전송하고, 부분 경로에 대한 연속 k-최근접 질의 처리를 병렬적으로 수행하기 때문이다. 아울러 MBP-CKNN 알고리즘은 한 번의 외부 확장으로 인접 셀에서의 POI를 검색하기 때문에, MCE-CKNN 알고리즘에 비해 검색 성능이 우수함을 나타낸다.

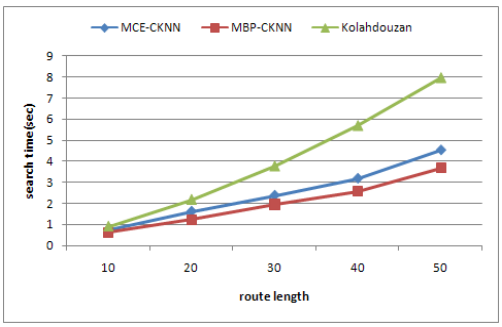

그림 5. 질의 경로의 길이 변화 따른 검색 성능

둘째, k 값의 변화에 따른 검색 시간을 측정한다. 이때, POI의 밀도는 0.01, 그리드 셀의 개수는 10*10 이다. 그림 6은 k 값의 변화에 따른 검색 성능을 나타낸다. k 가 1인 경우, Kolahdouzan의 방법은 보로노이 다이어그램을 이용하여 질의점에서 가장 가까운 POI를 검색하기 때문에, 검색 성능이 가장 우수함을 나타낸다. k 가 5 이상인 경우, 제안하는 연속 k-최근접 질의 처리 알고리즘은 부분 경로에 대한 연속 k-최근접 질의 처리를 병렬적으로

수행하기 때문에, 검색 성능이 보다 우수함을 나타낸다. MBP-CKNN 알고리즘은 셀 마다 경계점으로부터 가장 가까운 POI를 저장하기 때문에, 외부 확장에 따른 POI 검색 비용이 감소한다. 또한, 한 번의 외부 확장을 통해 경계점으로부터 가장 가까운 k 개의 POI를 메모리상에 유지하고, 이를 이용하여 추가적인 외부 확장을 수행하지 않고 다른 노드에서 k 개의 POI를 검색하기 때문에, 검색 성능이 가장 우수함을 나타낸다.

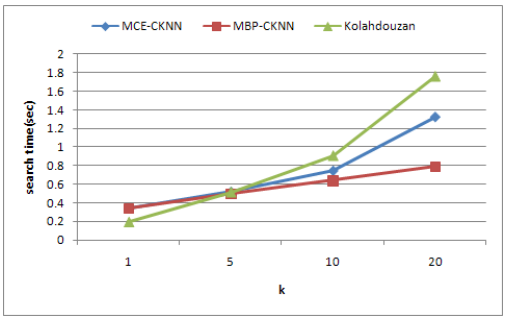

그림 6. k 값의 변화에 따른 검색 성능

셋째, POI 밀도의 변화에 따른 검색 성능을 비교한다. 이때, k 는 20, 그리드 셀의 개수는 10*10 이다. 그림 7 은 POI 밀도의 변화에 따른 검색 성능을 나타낸다. 세 방법 모두 POI 밀도가 증가할수록 검색 성능이 향상됨을 알 수 있다. 이는 POI 밀도가 증가할수록, k 개의 POI를 검색하기 위한 네트워크의 확장 횟수가 감소되며, 인접 셀에서 POI 탐색을 위한 외부 확장 횟수가 감소하기 때문이다.

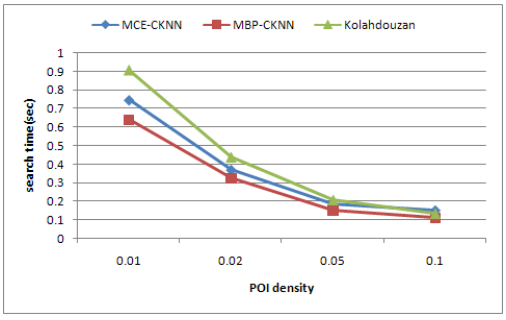

그림 7. POI 밀도의 변화에 따른 검색 성능

마지막으로, 분석적 성능 모델의 정확성을 분석하기 위해, 분석적 성능 모델에 따른 연속 k-최근접 질의 처리 알고리즘의 검색 성능 및 실제 구현을 통해 실험한 검색 성능을 비교한다. 이때 k 는 10, POI 밀도는 0.01, 그리드 셀의 개수는 10*10 이다. 표 1은 분석적 성능 모델 및 실험에 따른 검색 성능의 에러(error) 비율을 나타낸다.

표 1. 분석적 성능 모델 및 실험에 따른 검색 성능 비교

		10	20	30	40	50
Kolahdouzan	분석적 성능	0.841	2.022	3.499	5.278	7.335
	실험 성능	0.907	2.184	3.767	5.691	7.974
	error 비율	8.1 %	8.3 %	8.6 %	9.0 %	9.6 %
MCE-CKNN	분석적 성능	0.686	1.478	2.170	2.893	4.094
	실험 성능	0.746	1.612	2.374	3.180	4.531
	error 비율	8.1 %	8.3 %	8.6 %	9.0 %	9.6 %
MBP-CKNN	분석적 성능	0.587	1.127	1.782	2.355	3.349
	실험 성능	0.640	1.229	1.950	2.578	3.671
	error 비율	8.2 %	8.3 %	8.4 %	8.6 %	8.9 %

표에서 분석적 성능 모델과 실험에 따른 결과의 error 비율은, Kolahdouzan의 방법의 경우 8.1-9.6% 이며, MCE-CKNN 알고리즘의 경우 8.1-9.6%, MBP-CKNN 알고리즘은 8.2-8.9% 이다. 이때 경로의 길이가 증가할수록 분할점을 생성하기 위한 CPU 시간이 증가하기 때문에, error 비율이 완만하게 증가함을 알 수 있다. 그러나 세 방법 모두 error 비율이 9.6% 이하로써, 분석적 성능 모델이 정확함을 나타낸다.

6. 결론 및 향후 연구

본 논문에서는 공간 네트워크에서 대용량 이동객체를 위한 연속 k-최근접 질의처리를 위한 분산 처리 알고리즘 연구를 수행하였다. 이를 위해 이동 객체의 빈번한 업데이트로 인한 영향을 최소화할 수 있는 DS-GRID를 위한 MCE-CKNN 알고리즘 및 MBP-CKNN 알고리즘을 제안하고 이를 위한 분석적 성능 모델을 설계하였다. MCE-CKNN 알고리즘은 주어진 경로를 셀 단위로 분할하여 각 셀에서 MCE 알고리즘을 이용하여 질의 처리를 병렬적으로 수행함으로써, 검색 성능을 향상시켰다. 그러나 셀에서 경로를 구성하는 각 노드에 대한 MCE 알고리즘을 수행할 때, 인접 셀을 노드의 수만큼 반복적으로 방문하는 현상이 발생한다. 이를 해결하기 위해, MBP-CKNN 알고리즘은 경계점에서 가까운 POI를 미리 저장하고 한 번의 외부확장을 통해 POI를 검색함으로써 검색 성능을 향상시켰다. 아울러 기존 연구인 Kolahdouzan의 방법과의 성능 비교를 통해, 제안하는 방법이 15-53% 검색 성능이 우수함을 나타내었다.

향후 연구로는 제안하는 알고리즘을 실제 텔레매틱스

및 위치기반 서비스 응용에 적용하여 그 효율성을 입증하는 것이다.

참 고 문 헌

[1] 김영창, 김영진, 장재우, “대용량 이동객체의 위치정보 관리를 위한 S-GRID를 이용한 분산 그리드 기법”, 한국공간정보시스템학회 논문지, 10권, 4호, 2008, pp. 11-19.

[2] M. Kolahdouzan, C. Shahabi, “Voronoi-based k Nearest Neighbor Search for Spatial Network Databases”, In Proc. of VLDB, 2004, pages 840-851.

[3] X. Huang, C.S. Jensen, S. Saltenis, “The Islands Approach to Nearest Neighbor Querying in Spatial Networks”, In Proc. of SSTD 2005, LNCS 3633, 2005, pp. 73-90.

[4] 김용기, 니하드 카림 초우더리, 이현조, 장재우, “공간 네트워크 데이터베이스에서 실체화 기법을 이용한 범위 및 k-최근접 질의처리 알고리즘”, 한국공간정보시스템학회 논문지, 9권, 2호, 2007, pp. 67-79.

[5] X. Huang, C.S. Jensen, H. Lu, S. Saltenis, “S-GRID: A Versatile Approach to Efficient Query Processing in Spatial Networks”, In Proc. of SSTD 2007, LNCS 4605, 2007, pp. 93-111.

[6] M.R. Kolahdouzan, C. Shahabi, “Continuous K Nearest Neighbor Queries in Spatial Network Databases”, In Proc. of STDBM, 2004, pp. 33-40.

[7] Z. Song, N. Roussopoulos, “K-Nearest Neighbor Search for Moving Query Point”, In Proc. of SSTD, 2001, pp. 79-96.

[8] T. Brinkhoff, “A Framework for Generating Network-Based Moving Objects”, In Proc. of GeoInformatica, 2002, pp. 153-180.

김 영 창

2001년 전북대학교 컴퓨터공학과(공학사)
2003년 전북대학교 대학원 컴퓨터공학과 (공학석사)
2009년 전북대학교 대학원 컴퓨터공학과 (공학박사)
2009년~현재 한국전자통신연구원 선임연구원
관심분야 : 데이터 마이닝, 공간 데이터베이스, 공간 색인 구조, 질의처리 알고리즘

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
관심분야 : 공간 네트워크 데이터베이스, 위치 기반 서비스, 상황인식, 하부저장구조