순서정보및 Materialization 기법을 이용한 최근접 질의처리 알고리즘의 설계 및 구현

김영국[O] 김용기 김영창 장재우
전북대학교 컴퓨터공학과
{uksky[O], ykkim, yckim, jwchang}@dblab.chonbuk.ac.kr

Design and Implementation of an Order and Materialization-based K-Nearest Neighbors Query Processing Algorithm

Youngguk Kim[O] Yongki Kim Youngchang Kim Jaewoo Chang
Dept of Computer Engineering Chonbuk National University

요 약

최근 LBS(location-based service) 및 텔레매틱스(telematics) 응용의 효과적인 지원을 위해, 이상적인 유클리디언(Euclidean) 공간 대신, 실제 도로나 철도와 같은 공간 네트워크(network)를 고려한 연구가 활발하게 수행중이다. 본 논문에서는 공간 네트워크를 고려한 기존 k-최근접 질의 처리 알고리즘의 문제점을 제시하고, 공간 네트워크 데이터베이스에 보다 효율적인 새로운 k-최근접 질의 처리 알고리즘을 제안한다. 제안하는 질의처리 알고리즘은 순서정보 및 Materialization 기법에 근거하며 기존 방법의 검색 성능을 향상시킨 방법이다. 마지막으로 제안하는 k-최근접 알고리즘을 기존의 알고리즘과 성능 비교를 수행한다.

1. 서 론

최근 LBS(location-based service) 및 텔레매틱스(telematics) 응용의 효과적인 지원을 위해, 이상적인 유클리디언(Euclidean) 공간 대신, 실제 도로나 철도와 같은 공간 네트워크(network)를 고려한 연구가 활발하게 수행 중에 있다[1,2,3,4,5]. 이러한 공간 네트워크 데이터베이스 연구는 일반 공간 데이터베이스의 연구와 마찬가지로 크게 3가지 주제로 요약된다. 즉, 공간 네트워크를 위한 데이터 모델, 질의 처리 알고리즘, 마지막으로 공간 네트워크 데이터베이스를 위한 저장 시스템이다.

본 논문에서는 위의 3가지 주제 가운데, 공간 네트워크 데이터베이스에서의 효율적인 질의처리 알고리즘을 설계하고자 한다. 공간 네트워크는 도로나 철도와 같은 이미 정해진 공간 네트워크 상에서 객체의 이동이 이루어지기 때문에, 공간 네트워크 상에서의 질의처리 알고리즘은, 이상적인 유클리디언(Euclidean) 공간을 가정한 기존의 질의처리 알고리즘과는 매우 다르다. 최근 공간 네트워크 상에서의 k-최근접 질의 처리 알고리즘으로, 유클리디언 공간에서의 k-최근접 알고리즘을 확장하는 방법(IER)과 네트워크를 확장하는 방법(INE)이 제안되었으나 [5], 이 방법들은 응용 환경에 따라서 다수의 디스크 I/O 나 불필요한 연산 등이 존재한다. 따라서 본 논문에서는 네트워크 확장 방법에 추가적으로 Materialization 기법 및 순서 정보 개념을 도입하여 보다 효율적으로 k-최근접 질의를 처리하는 알고리즘을 제안한다. 아울러 제안하는 알고리즘을 기존 k-최근접 알고리즘과 성능비교를 수행한다.

논문의 구성은 다음과 같다. 2장에서는 관련 연구를 소개하고, 3장에서는 본 논문에서 설계한 k-최근접 질의처리 알고리즘을 제시한다. 4장에서는 제안하는 알고리즘과 기존 알고리즘과의 성능 비교를 수행하고, 마지막으로 5장에서는 결론 및 향후연구를 제시한다.

본 연구는 대학 IT 연구센터 육성·지원사업의 연구결과로 수행되었음

2. 관련연구

공간 네트워크 데이터베이스를 위한 대표적인 질의처리 알고리즘으로, HKUST에서는 공간 네트워크 및 유클리디언 공간을 합성한 구조를 제안하였다[5]. 아울러, 이 구조를 기반으로 k-최근접 질의처리 알고리즘으로 IER(Incremental Euclidean Restriction) 및 INE(Incremental Network Expansion) 를 제안하였다. HKUST의 성능평가 결과, INE 이 방법이 훨씬 우수하므로, 여기서는 INE 방법만을 설명한다.

INE 방법은 질의가 속한 에지(edge)를 탐색하여, 이 에지를 시작으로 에지상에 존재하는 k개의 최근접점을 찾을 때까지 에지를 큐(queue)에 추가하면서 네트워크를 확장해가는 방법이다. 이를 통해 얻어지는 k-최근접점 POI(Point of Interest)들은 그들의 거리가 공간 네트워크상의 거리를 의미한다. 새로이 추가되는 에지까지의 거리가 k번째 최근접점 네트워크 거리보다 커지면 알고리즘은 종료된다. 그러나 INE 알고리즘은 기본적으로 Dijkstra 알고리즘을 사용하여 최근접점을 찾기 때문에, 질의점에서 먼 거리에 있는 POI를 찾거나 k 값이 증가하면 알고리즘의 특성상 전체적인 검색 시간이 증가하게 된다.

3. 제안하는 알고리즘의 설계

INE에서 노드까지의 거리 계산은 각 노드의 정보를 디스크로부터 읽고 질의점에서 각 노드까지의 거리를 계산해야 하므로, 디스크 I/O의 증가와 함께 Cpu 시간의 증가를 야기한다. 따라서 본 논문에서는 검색에 필요한 한 노드로부터 다른 모든 노드까지의 거리를 미리 계산하여 저장하는 Materialization 기법을 이용한다. 또한, 계산된 거리를 사용하여 각 노드로부터

다음으로 가까운 노드를 알 수 있는 순서정보를 함께 저장함으로써, 한 노드에서 가장 가까이에 있는 노드, 또는 n 번째에 있는 노드를 검색할 수 있다. 본 논문에서 제안하는 알고리즘(OMK: Order & Materialization-based K-nearest Neighbors Algorithm) 은 순서정보가 포함된 Materialization 파일을 이용하여 K-최근접 질의를 효율적으로 처리할 수 있다.

3.1 Materialization 파일의 구조

본 논문에서 제시하는 Materialization 파일의 한 레코드의 구성은 (그림1)과 같다. 여기서 Start Node ID는 기준점의 시작 노드식별자를, Destination Node ID는 도착지점 노드식별자를, Distance는 Start Node와 Destination Node의 거리를, 마지막으로 Next rank ID는 Destination Node 다음으로 Start Node로부터 가까운 노드 식별자를 나타낸다.

Start Node ID	Destination Node ID	Distance (float : 4Byte)	Next Rank ID (int : 4 Byte)
1	1	0	2
	2	3424.23	169999
	...	...	...
	n	34323.82	2342
2	1	23434.234	46
	2	0	56
	...	...	...
	n	2568.56	45
...	...	...	...
n	1	3567.8	36
	2	5894.24	467
	...	...	...
	n	0	26

그림1. Materialization 파일의 레코드 구조

Materialization 파일을 사용함으로써 다음과 같은 장점이 있다. 첫째, 노드식별자만을 가지고 노드와 노드 사이의 최소 거리를 계산 없이 바로 구할 수 있다.둘째, 순서정보를 포함하고 있으므로 어떤 노드에서라도 가장 가까이에 있는 노드 또는 n 번째에 있는 노드를 바로 알 수 있다. 이러한 장점으로 인하여 노드의 정보를 읽어 들이는 횟수를 줄이기 때문에, 디스크 I/O 수가 감소된다. 아울러 노드간의 거리계산을 줄이므로 질의 처리를 위한 CPU 시간을 줄일 수 있다. 이것을 통해 전체검색시간이 감소한다. 본 논문에서 제안하는 순서정보를 지닌 Materialization 파일을 이용한 OMK 알고리즘은 다음 단계에서노드의 확장할 노드식별자 및 노드까지의 거리를 계산 없이 알 수 있으므로, 기존 INE 알고리즘보다 디스크 I/O 수 및 Cpu 시간을 줄일 수 있다.

3.2 OMK 알고리즘

```
01.  Algorithm OMK (q, k)
02.  // q is the query point
03.  // EES is the Extra Edge Set
04.  // result is the POI(Point Of Interesting) Set
05.  // ns is the Node length Store, Materialization File
06.  // hn is the History Node
07.  result = ∅;
08.  hn = ∅;
09.  dmax = ∞ ;
10.  result= result ∪ find_poi(q.ni,q.nj);
11.  ns = load_ns(ni, nj);
12.  hn = hn ∪ ni;
13.  hn = hn ∪ nj;
14.  while(nsx-1.length < dmax) {
15.     if(count(result) >= k)  {
16.        result = sort_kcut(result, k);
17.        dmax = max(result);   }
18.     if(hn⊇nsx)    {
19.        hn = hn ∪ nsx;
20.        foreach(nsx)  {
21.           if(hn⊇nsx.nj) {
22.           result = result ∪ find_poi(nsx.ni, nsx.nj);
23.              EES = EES - nsx.edge_idj; }
24.           else       {
25.              EES = EES ∪ nsx.edge_idj;}}}
26.     nsx = nsx+1;     }
27.  foreach(EES){
28.     result = result ∪ find_poi(EES); }
29.  result = sort_kcut(result, k);
30.  return result;;
31.  End OMK
```

그림2. Order Materialization K-nn Algorithm

본 논문에서 제안하는 OMK 알고리즘은 (그림2)와 같다. 첫째, 질의점을 포함하는 에지를 탐색하여 얻어진 POI 집합을 후보 결과 집합에 넣는다. 탐색된 에지의 양 끝 노드는 HN(History Node)에 포함시킨다. 즉, HN에는 탐색된 노드들의 정보를 포함한다. 둘째, 질의점이 포함되어 있는 에지의 양 끝 노드로부터 모든 노드까지의 거리와 순서정보를 Materialization 파일로부터 탐색한다. Materialization 파일로부터 질의점으로 부터 가장 가까이에 있는 노드부터 검사한다. 만약, 그 노드가 HN에 포함되어 있으면, Materialization 파일에서 질의점에서 다음으로 가까운 노드를 읽고, 그렇지 않으면 그 노드를 HN에 포함시키고, 노드의 정보를 읽어서 인접노드 리스트를 생성한다. 인접노드 중 HN에 포함되어 있는 노드와 Materialization 파일에서 바로 전에 읽었던 노드로 이루어진 에지를 탐색하여 POI 집합을 구하고, 이를 후보 결과 집합에 포함시킨다. 인접노드 리스트중 HN에 포함되지 않는 노드와 Materialization 파일에서 바로 전에 읽었던 노드로 이루어진 에지는 나중에 탐색하기 위해서 EES(Extra Edge Set)에 넣는다. 셋째, 후보 결과 집합의 개수가 k보다 크거나 같으면 질의점으로 부터 거리가 작은 순으로 정렬 한 후, 상위 k개만을 선택하여 k번째 POI와 질의점까지의 거리를 dmax로 정의한다. 넷째, Materialization 파일에서 바로 전에 읽었던 노드와 질의점까지의 거리가 dmax 보다 크거나 같게 되면, EES에 들어 있는 에지상의 모든 POI를 검색하여 후보 결과 집합에 넣는다. 마지막으로 후보 결과 집합을 정렬한 후, 상위 k개를 최종 결과 집합으로 전달한다.

3.3 OMK 알고리즘의 예시

(그림 3)에서 "q에 가까이 있는 최근접 POI를 찾아라" 라는 질의는 본 논문에서 제안한 OMK 알고리즘에 따라 다음과 같이 수행된다. 첫째, 질의 q가 속한 에지 e(n2,n3)을 탐색하고 n2, n3 노드의 Materialization 파일을 통해 큐를 생성한다. 즉, <(n3,5),(n4,7),(n2,8),(n1,12),(n5,13),(n6,17)> 이다, e(n2,n3)은 탐색을 했기 때문에 초기 HN의 구성은 <n2,n3>가 된다. 둘째, Materialization 파일에서 질의점으로 부터 가장 가까이에 있는 노드가 n3이지만, HN에 이미 포함되어 있으므로, 그 다음 노드인 n4를 읽는다. n4는 HN에 포함되어 있지 않으므로 HN에 포함시킨다. n4의 인접 노드는 <n3, n5>이다. 인접노드중 HN에 포함되어 있는 것은 n3 이므로, e(n4,n3)을

탐색하고 e(n4,n5)는 EES에 넣는다. 셋째, 아직 POI가 검색되지 않았기 때문에, Materialization 파일에서 다음 노드를 읽고(n2), 이것이 HN에 포함되어 있기 때문에, 그 다음 노드인 n1을 읽는다. n1는 HN에 포함되어 있지 않으므로 HN에 포함시킨다. 넷째, n1의 인접 노드는 <n2,n6>이다. 인접 노드중 HN에 포함되지 않은 것은 n6이므로, e(n1,n6)를 EES에 넣고 e(n1,n2)를 탐색하여서 POI p2를 찾는다. 이때 dmax 를 q에서 p2 까지의 거리인 10으로 설정한다. 마지막으로, 현재 EES에서 에지 e(n4,n5),e(n1,n2) 를 탐색하면 POI p1를 찾는다. 질의점에서 n1까지의 거리가 12 이므로 이는 dmax 보다 크다. 따라서 최종 결과 POI 집합은 {(p2,10)}이 된다.

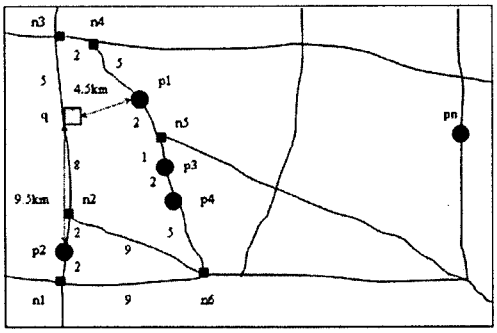

그림 3. INE를 이용한 k-최근접 질의 처리

4. 알고리즘의 구현 및 성능 평가

성능평가를 위해 본 논문에서 제안하는 알고리즘을 Memory 2GB, CPU 2.4GHz를 지닌 Windows Server 2003 Enterprise에서 Visual C++ 7.1을 사용하여 구현하였다. 지도 데이터는 실제 샌프란시스코 만 지도를 사용하였고 [6], 전체 노드 수는 175,343개, 전체 에지 수는 223,199개로 구성되어 있다. 아울러, POI는 RunTime21[7] 알고리즘을 사용하여 임의로 생성한 10846개를 사용하였고, 질의점은 임의로 1000개를 주어서 사용하였다. 마지막으로 Materialization 파일의 크기는 229GB이며, 알고리즘의 구현을 위해 이미 개발된 공간 네트워크를 위한 저장/색인 구조를 사용하였다[8]. (표 1)은 INE와 본 논문에서 제안하는 OMK를 k-최근접 (k=1~200) 알고리즘을 수행하는 동안의 디스크 I/O 횟수, CPU 시간, 전체검색 시간(Wall Time)을 측정한 것이다.

(표 1). INE 와 OMK의 성능 비교

	K	Disk I/O(횟수)	CPU Time(초)	Wall Time(초)
INE	1	26.22	0.015933	0.048604
	5	761.82	0.019129	0.056423
	10	1764.99	0.026642	0.069623
	25	4697.95	0.056587	0.119906
	50	9687.69	0.121338	0.220764
	100	19766.13	0.309421	0.506020
	200	39924.36	0.822166	1.276582
OMK	1	9.82	0.024853	0.066881
	5	281.39	0.024853	0.066981
	10	597.49	0.025373	0.068834
	25	1483.83	0.028617	0.077774
	50	2953.98	0.030911	0.092815
	100	5862.61	0.040359	0.124248
	200	11632.16	0.058806	0.189779

k=1일 때 전체 검색시간은 INE의 경우 0.0486초이고, OMK는 0.0668초이다. 이때, OMK가 INE에 비해 성능이 저하되는 이유는, 초기 Materialization 파일을 로드하는 시간이 추가되기 때문이다. 하지만 k=10일 때 INE의 전체 검색시간은 0.0696초이고, OMK는 0.0688초가 되며, k값이 10일 때부터 OMK가 INE에 비해서 성능이 우수함을 알 수 있다. (그림 4)는 INE와 OMK의 전체 검색시간을 k값에 따라서 그래프로 표시한 것이다. INE와 OMK 모두 k값이 증가 할수록 전체 검색시간이 증가함을 알 수 있다. 하지만 본 논문에서 제안하는 OMK는 INE 보다 훨씬 완만하게 증가함을 볼 수 있다. k값이 200일 때, OMK는 INE에 비해서 약 6배의 성능 개선이 이루어짐을 알 수 있다.

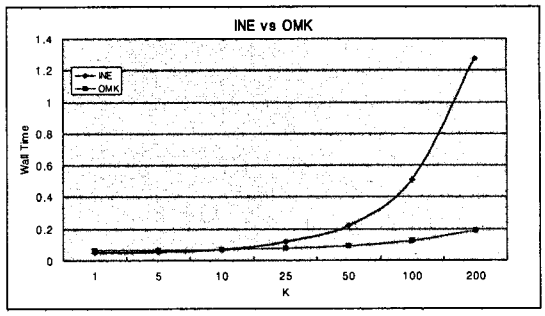

그림4. INE 와 OMK의 전체검색시간 비교

5. 결론 및 향후 연구

본 논문에서는 공간 네트워크 데이터베이스를 위한 기존 k-최근접 질의 알고리즘인 INE를, Materialization 기법과 순서 정보를 이용하여 확장한 OMK 알고리즘을 제안하였다. 또한, 성능 평가를 통해, OMK 는 INE보다 전체 검색 시간 측면에서 k=10 이상에서 k에 비례하여 우수함을 보였다.

향후 연구로는 k-최근접 질의 처리 알고리즘뿐만 아니라, 다른 다양한 공간 질의 처리 알고리즘을 Materialization 기법을 사용하여 설계하고 성능평가를 수행하는 것이다.

[참고문헌]

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

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

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

[4] 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,, pp 255-273, 2003.

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

[6] http://www.fh-oow.de./institute/iapg

[7] T. Brinkhoff, "A Framework for Generation Network - Based Moving Objects" , GeoInformat

[8] 강홍민, 장재우 "공간 네트워크 데이터베이스를 위한 저장 및 색인 구조의 설계" 한국정보과학회 가을 학술발표논문집 제31권 제2호 pp133-136, 2004