공간 네트워크 데이터베이스에서 POI 기반 실체화 기법을 이용한 Closest Pairs 및 e-distance 조인 질의처리 알고리즘†

Closest Pairs and e-distance Join Query Processing Algorithms using a POI-based Materialization Technique in Spatial Network Databases

김용기* / Yong-Ki Kim, **장재우**** / Jae-Woo Chang

요 약

최근 LBS(location-based service) 및 텔레매틱스(telematics) 응용의 효율적인 지원을 위해, 기존 유클리디언(Euclidean) 공간 대신, 실제 도로나 철도와 같은 공간 네트워크(network)를 고려한 다수의 연구가 수행되었다. 그러나 Closest Pairs 질의 및 e-distance 조인 질의는, 하나의 POI(Point Of Interest)를 다루는 대신 POI 집합에 대하여 질의처리를 수행하기 때문에 매우 비용이 많이 든다. 아울러, k 값 및 범위의 증가에 따라 질의처리에 필요한 노드 검색 및 거리 계산의 비용이 매우 크게 증가한다. 따라서 본 논문에서는 공간 네트워크를 위한 효율적인 Closest Pairs 질의 및 e-distance 조인 질의 처리를 위해, POI 기반의 실체화 기법을 이용한 효율적인 질의처리 알고리즘을 제안한다. 아울러 기존 질의처리 알고리즘과의 성능 비교를 통하여 제안하는 알고리즘이 검색 성능이 우수함을 보인다.

Abstract

Recently, many studies on query processing algorithms has been done for spatial networks, such as roads and railways, instead of Euclidean spaces, in order to efficiently support LBS(location-based service) and Telematics applications. However, both a closest pairs query and an e-distance join query require a very high cost in query processing because they can be answered by processing a set of POIs, instead of a single POI. Nevertheless, the query processing cost for closest pairs and e-distance join queries is rapidly increased as the number of k (or the length of radius) is increased. Therefore, we propose both a closest pairs query processing algorithm and an e-distance join query processing algorithm using a POI-based materialization technique so that we can process

† 이 논문은 교육인적자원부, 산업자원부, 노동부의 출연금으로 수행한 최우수실험실지원사업의 연구결과이며, 아울러 이 연구에 참여한 연구자는 2단계 BK21 사업의 지원비를 받았음

■ 논문접수 : 2007.10.29 ■ 심사완료 : 2007.12.12

* 전북대학교 대학원 컴퓨터공학과 박사과정 (ykkim@dblab.chonbuk.ac.kr)

** 교신저자 전북대학교 컴퓨터공학과 교수 (jwchang@chonbuk.ac.kr)

closest pairs and e-distance join queries in an efficient way. In addition, we show the retrieval efficiency of the proposed algorithms by making a performance comparison of the conventional algorithms.

주요어 : 공간 네트워크 데이터베이스, 질의처리 알고리즘, closest pairs 질의, e-distance 조인 질의
Keyword : Spatial network database, query processing algorithms, closest pairs query, e-distance join query

1. 서 론

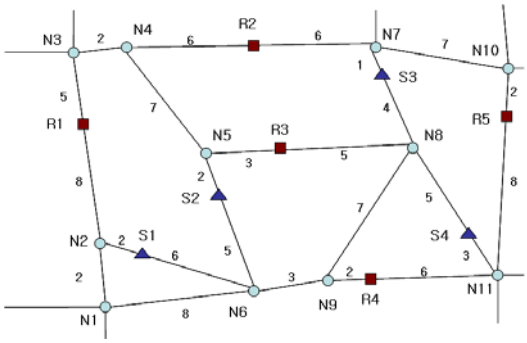
최근 사용자의 위치를 기반으로 한 LBS(location-based service) 및 텔레매틱스(telematics) 응용이 많이 활성화 되어 있다. 이를 효과적으로 지원하기 위해, 기존 유클리디언(euclidean) 공간 대신, 실제 도로나 철도와 같은 공간 네트워크(spatial network)를 고려한 연구가 활발하게 수행 중에 있다[1,2,3,4,5,6,7]. 이러한 공간 네트워크 데이터베이스(spatial network databases) 연구는 크게 3가지 주제로 요약된다. 즉, 공간 네트워크를 위한 데이터 모델, 질의 처리 알고리즘, 마지막으로 공간 네트워크 데이터베이스를 위한 저장 및 색인 구조이다.

첫째, 공간 네트워크 데이터베이스에서 데이터 모델에 관한 대표적인 연구는 덴마크의 Aalborg 대학의 연구가 존재한다[2,8]. Aalborg 대학의 연구에서는 네트워크 내의 이동 객체를 위한 데이터 모델링을 제시하였다. 여기서는 도로 네트워크(road network) 및 고정/이동 객체의 2차원 표현 및 그래프 표현 방법을 제시하고, 2차원 표현을 그래프 표현으로 변환하는 방법을 제시하고 있다[2]. 둘째, 공간 네트워크 데이터베이스에서 질의처리 알고리즘에 관한 연구는 하나의 POI(Point Of Interest) 타입을 고려한 k-최근접 질의, 범위 질의 및 두 개 이상의 POI 타입을 고려한 Closest Pairs 질의 및 e-distance 조인 질의처리 알고리즘에 관한 연구가 존재한다[5,6]. 홍콩 과학기술대학(HongKong University of Science and Technology)에서는 유클리디언 공간 확장 기법(Euclidean Restriction Method)과 네트워크 확장 기법(Network Expansion Method)을 기반으로 하여 k-최근접 질의, 범위 질의, Closest Pairs 질의 및 e-distance 조인 질의처리 알고리즘을 제안하였다[6]. 마지막으로, 공간 네트워크 데이터의 효율적인 색인 및 저장 시스템의 연구는 공간 네트워크를 그래프 형태로 변환한 후 저장하는 연구들이 수행되었다[6,9]. 대표적인 연구로써 미네소타 대학의 연구에서는, 공간 네트워크를 그래프 형태로 변환하여 표현하는 저장 기법을 제시하였다[9]. 아울러 이렇게 변환된 그래프를 Z-order 로 표현하여 연결에 따른 클러스터를 형성한 노드들에 대해 B+-트리를 구성함으로써 노드 식별자를 통한 빠른 접근을 가능하게 한다.

본 논문에서는 위의 3가지 주제 가운데, 공간 네트워크 데이터베이스에서 비용이 많이 드는 질의처리 알고리즘인 Closest Pairs 질의 및 e-distance 조인 질의처리 알고리즘을 다룬다. 이러한 질의는 모바일 컴퓨팅 (Mobile Computing), 위치 기반 서비스(Location Based Service) 및 텔레매틱스(Telematics)를 기반으로 한 최근 응용에 매우 중요한 역할을 한다. 따라서 이러한 질의를 효율적으로 처리하는 알고리즘이 필수적이다.

먼저 Closest Pairs 질의처리 알고리즘은 한 타입의 모든 POI에 대하여 다른 타입의 POI에 대한 가장 가까운 k개의 POI 쌍을 찾는 질의이다. 예를 들면, "호텔에서 레스토랑까지의 거리가 가까운 5개의 쌍을 찾아라"와 같은 질의이다. <그림 1>에서 R 셋에 속하는 POI들에 대하여 가까운 3개의 S 셋에 속하는 POI 쌍을 찾고자 한다면, R셋에 속하는 5개의 POI에 대하여 S 셋에 속하는 4개의 POI에 대한 거리를 계산하여, 각각의 거리가 5, 7, 9

인 3개의 POI 쌍, (R3, S2), (R2, S3), (R4, S4)를 검색하는 질의이다. 한편, e-distance 조인 질의처리 알고리즘은 한 타입의 모든 POI에 대하여 다른 타입의 POI에 대한 거리가 e 이내인 모든 POI 쌍을 찾는 질의이다. 예를 들면, "식당에서 극장까지의 거리가 3km이내인 쌍을 찾아라"와 같은 질의이다. <그림 1>에서 범위 10이내인 POI 쌍을 찾고자 할 때, 질의 처리결과는 (R3, S2, 5), (R2, S3, 7), (R4, S4, 9), (R1, S1, 10), (R4, S2, 10), (R5, S3, 10) 이 된다.



<그림 1> Closest Pairs 질의 및 e-distance 조인 질의처리 알고리즘의 예

Closest Pairs 질의 및 e-distance 조인 질의처리 알고리즘은 두 가지 셋을 고려해야 하기 때문에 k-최근접 질의 및 범위 질의에 비해 복잡할 뿐만 아니라 처리 비용이 많이 든다. 그 이유는 Closest Pairs 질의 및 e-distance 조인 질의는 한 질의점에서 수행하는 것이 아니라 POI 집합내의 모든 POI에 대하여 질의를 수행하기 때문에 k 값 및 범위의 증가에 따라 검색에 필요한 노드 탐색 및 거리 계산의 비용이 기하급수적으로 증가한다. 따라서 본 논문에서는 공간 네트워크를 위한 효율적인 Closest Pairs 질의 및 e-distance 조인 질의처리를 위해, POI 정보 기반의 실체화 기법을 이용한 효율적인 질의처리 알고리즘을 제안한다. 아울러 제안하는 알고리즘의 우수성을 보이기 위해 기존 질의처리 알고리즘과의 성능비교를 수행한다.

논문의 구성은 다음과 같다. 2장에서는 관련 연구를 소개하고, 3장에서는 본 논문에서 제안하는 POI 정보 기반의 실체화 기법을 제시한다. 4장에서는 POI 정보 기반의 실체화 기법을 이용한 Closest Pairs 질의 및 e-distance 조인 질의처리 알고리즘을 제안한다. 5장에서는 제안하는 알고리즘과 기존 알고리즘과의 성능평가를 수행하고, 마지막으로 6장에서는 결론 및 향후연구를 제시한다.

2. 관련 연구

공간 네트워크 데이터베이스를 위한 대표적인 질의처리 알고리즘으로 k-최근접 질의, 범위 질의, Closest Pairs 질의 및 e-distance 조인 질의처리 알고리즘이 존재한다. 이 중에서 k-최근접 질의는 질의점에서 가장 가까운 k개의 POI를 찾는 질의이며, 범위 질의는 질의점에서 주어진 범위 이내의 POI를 찾는 질의이다[5,6]. 반면에, Closest Pairs 질의는 같은 타입의 POI에서 다른 타입의 POI까지의 가장 가까운 k개의 POI 쌍을 찾는 질의이며, e-distance 조인 질의는 두 타입이 일정 범위 이내에 존재하는 POI 쌍을 찾는 질의이다[6]. 따라서 Closest Pairs 질의 및 e-distance 조인 질의는 POI의 타입을 고려한 k-최근접 질의, 범위 질의를 각각 확장한 질의라고 할 수 있다.

Closest Pairs 질의 및 e-distance 조인 질의의 대표적인 연구로는 홍콩 과학기술 대학의 연구가 존재하며[6], 이 연구에서는 유클리디언 공간을 확장하는 기법과 공간 네트워크를 확장하는 기법을 제시하였다. 제시된 두 기법을 기반으로 Closest Pairs 질의처리 알고리즘으로써 CPER (Closest Pairs Euclidean Restriction) 및 CPNE (Closest Pairs Network Expansion) 을 제안하였고, e-distance 조인 질의처리 알고리즘으로써 JER (e-distance Join Euclidean Restriction) 및 JNE (e-distance Join Network Expansion)를 제안하였다. 첫째, CPER 기법은 두 타입의 POI를 달리 저장한 각각의 R-tree를 사용하여 유클리디언 거리로써 k 개의 최근접점을 검색한다. 검색된 결과를 네트워크 거리의 계산을 통해 정렬한 후에,

후보 집합으로 저장과 동시에 상계값(upper bound)을 설정한다. 그리고 유클리디언 거리로 다음으로 가까운 POI 쌍을 검색하여 네트워크 거리를 계산한 후 상계값보다 작으면 결과 후보집합에 포함시킨다. 검색해야할 POI 쌍의 유클리디언 거리가 상계값보다 크거나 같으면 알고리즘은 종료한다. 둘째, CPNE 기법은 네트워크의 특성을 이용한 네트워크 확장 방법으로 기준이 되는 타입의 첫 번째 POI로부터 k-최근접 질의처리를 수행한 후, 상계값을 설정한다. 설정된 상계값을 이용하여 다음 POI에서 상계값보다 실제 거리가 짧은 POI를 탐색하여 상계값을 갱신하며, 이를 마지막 POI까지 반복적으로 수행하는 방법이다. 셋째, JER 기법은 유클리디언 거리로써 각 POI 타입의 R-tree를 이용하여 조인 연산을 수행한 후, 조인 결과를 거리에 근거하여 정렬하여 두 POI의 유클리디언 거리가 범위 e이내의 후보 셋을 구한다. 후보 셋의 모든 POI 쌍을 대상으로 실제 거리를 최단 거리 알고리즘을 사용하여 거리를 구하고 구한 거리가 범위 e 이내인 POI의 쌍을 구하는 알고리즘이다. 마지막으로, JNE 기법은 POI로부터 범위 e내의 질의가 속한 에지(edge)를 탐색하여, 그 에지를 대상으로 에지상에 존재하는 POI를 찾는 방법이다. 이 방법은 CPNE와 마찬가지로 기준이 되는 POI들을 대상으로 모두 수행함으로써 결과 집합을 얻는다.

그러나 홍콩 과학기술 대학의 Closest Pairs 질의처리 알고리즘 및 e-distance 조인 질의처리 알고리즘들은 다음과 같은 단점을 지니고 있다. 먼저 유클리디언 공간을 이용하여 Closest Pairs 질의 및 e-distance 조인 질의를 처리하는 CPER 및 JER 알고리즘은 유클리디언 공간에서 k개의 POI 쌍을 구하거나 범위 e이내의 POI 쌍을 구한 후, 실제 거리를 구해야 한다. 따라서 POI가 저장된 R-tree를 탐색함과 동시에 모든 후보 POI 쌍의 실제 거리를 구해야 하므로, k 값 및 범위가 증가할수록 디스크 I/O 증가 및 CPU 시간 증가의 오버헤드가 존재한다. 또한 네트워크를 확장하는 기법인 CPNE 및 JNE는 기준이 되는 타입의 POI 전체에 대하여 k-최근접 알고리즘 및 범위 알고리즘을 수행하여야 하므로 성능이 좋지 못하다.

아울러 질의를 수행하는 방법은 크게 2가지로 나눌 수 있다. 첫째, 동적 연산 기법을 사용하여 질의처리를 수행하는 방법이다[6]. 이는 질의를 수행하면서 동적으로 메모리 할당 및 네트워크를 탐색하여 질의를 수행한다. 이는 갱신 비용이 최소화 되며, 부가적인 저장 공간이 필요치 않다. 그러나 네트워크를 탐색하여 질의를 수행하므로 질의응답 속도가 느리다. 둘째, 질의처리를 수행하기 전에 네트워크 정보를 미리 계산하여 저장함으로써 질의응답 시간을 최소화 하는 pre-computation 기법이 있다. 이는 미리 네트워크를 탐색하여 질의 수행에 필요한 정보들을 미리 연산하여 저장함으로써 연산 시간을 줄이고 보다 빠른 질의응답 속도를 제공한다. 이러한 pre-computation 기법에는 보로노이 다이어그램을 이용한 기법과 노드에서 노드까지의 거리를 미리 계산하는 실체화 기법이 있다. 그러나 보로노이 다이어그램을 이용한 기법은 각각의 POI에 대하여 보로노이 영역(Voronoi Polygon)으로 분할하여 질의를 수행하는 방법이다[5]. 보로노이 영역은 자신이 가지고 있는 POI를 다른 POI보다 가까운 영역을 나타낸다. 그러나 이는 여러 POI를 고려하지 못하는 단점을 지니고 있다. 반면 실체화 기법은 POI가 가지고 있는 양 노드에서 다른 POI가 가지고 있는 양 노드까지의 거리를 미리 계산하여 저장하고 있으므로 POI에서 다른 POI까지의 거리를 쉽게 계산하기 쉬울 뿐만 아니라, POI 타입에 대해 영향을 받지 않는 장점을 지니고 있다[10]. 반면, 기존 노드-노드 실체화 기법은 질의점이 하나일 때는 하나의 실체화 파일을 읽음으로써 효율적으로 처리할 수 있으나, Closest Pairs 질의 및 e-distance 조인 질의를 처리하기 위해서는 POI를 포함하고 있는 여러 실체화 파일을 읽어야 한다. 아울러 모든 노드간의 거리를 포함하고 있으므로 질의를 수행하기 위한 필요치 않는 정보를 포함하고 있어 질의 수행의 최적화를 이루지 못하는 단점을 지니고 있다.

3. POI 정보 기반의 실체화 기법

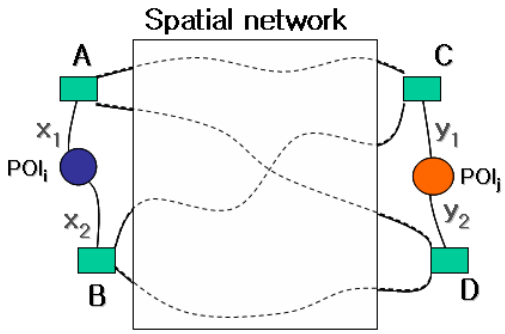
Pre-computation 기법으로써 노드-노드 실체화 파일은 미리 네트워크를 탐색하여 질의 수행에 필요한 정보들을 미리 연산하여 저장함으로써 질의 수행중의 연산 시간을 줄이고 빠른 질의응답을 제공한다[10]. 본 논문에서는 기존 노드-노드 실체화 파일을 Closest Pairs 질의 및 e-distance 조인 질의를 수행하는데 있어 문제점을 파악하고, 이를 효율적으로 처리할 수 있는 POI를 고려한 새로운 실체화 기법을 제안한다. 제안하는 실체화 기법은 POI 타입의 개수에 근거하여, 첫째 범용적인 경우에 효율적인 POI 타입 기반 실체화 파일 구성 방법과, 둘째 POI 타입의 개수가 매우 적을 경우에 효율적인 POI 타입 쌍 기반 실체화 파일 구성 방법을 제시한다.

3.1 기존 노드-노드 실체화 파일의 문제점

노드-노드 실체화 파일은 네트워크상에서 노드와 노드간의 실제 거리를 미리 계산하여 저장한다. 즉, POI를 포함하고 있는 에지의 양 노드에서 다른 POI를 구성하고 있는 양 노드까지의 거리를 미리 계산하여 저장한다. 이를 통해, 한 POI에서 다른 POI까지의 거리를 쉽게 계산하기 쉬울 뿐만 아니라, POI 타입에 대해 영향을 받지 않는다. 다음 정의 1은 노드-노드 실체화 파일을 이용하여 한 POI에서 다른 POI까지의 거리를 구하는 계산식이다[10]. 여기서 $ND(POI_i, POI_j)$ 는 두 POI 간의 네트워크 거리를 의미하며, 각 POI를 포함하는 에지를 이용한 실제 거리는 <그림 2> 에서 제시한다.

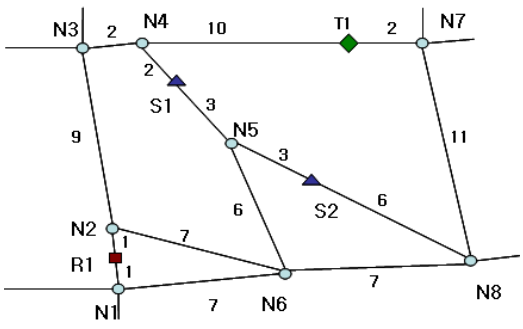
정의 1. 두 POI의 네트워크 거리 ND(POI, POI)

$$ND(POI_i, POI_j) = MIN(ND(A,C)+x1+y1,\ ND(A,D)+x1+y2,\ ND(B,C)+x2+y1,\ ND(B,D)+x2+y2)$$



<그림 2> POI간의 거리계산의 예

<그림 3>은 POI 를 포함하는 노드 및 에지로 구성된 공간 네트워크 예제를 나타내며, 여기서 노드의 개수는 8개이고, POI 개수는 전체 4개로, R 타입 1개, S 타입 2개, T 타입 1개로 구성된다. <그림 4>는 공간 네트워크 예제에 대한 노드-노드 실체화 파일을 나타낸다. 노드-노드 실체화 파일을 이용하여 <그림 3>에서 POI R1과 S1간의 거리는 정의 1을 이용하여 쉽게 구할 수 있다. 즉, 실체화 파일을 이용한 POI R1과 S1간의 거리는 MIN(13+1+2, 13+2+3, 11+1+2, 13+1+3) 값을 계산하여 14 임을 알 수 있다. 그러나 노드-노드 실체화 파일은 모든 노드-노드간의 실제 네트워크 거리를 저장하고 있기 때문에, 노드 3과 6의 경우처럼 실제 질의 수행에 전혀 쓰이지 않는 불필요한 노드를 포함하고 있다. 아울러, Closest Pairs 질의 및 e-distance 질의를 수행하기 위해서는 한 개의 질의점을 고려하는 것이 아니라 여러 개의 질의점을 고려하여야 하므로, 실체화 파일 전체가 메모리에 적재되어 있지 않은 대부분의 경우에는 실체화 파일을 다수 접근해야 하는 단점을 가지고 있다. 따라서 Closest Pairs 질의 및 e-distance 조인 질의를 효율적으로 처리하기 위해서는 POI 정보를 고려한 실체화 파일에 관한 연구가 필요하다.



<그림 3> 공간 네트워크 예제

S.. Node	D.. Node	Distance
1	2	2
1	3	11
1	4	13
1	5	13
1	6	7
1	7	25
1	8	14
5	1	13
5	2	13
5	3	7
5	4	5
5	6	6
5	7	17
5	8	9

S.. Node	D.. Node	Distance
2	1	2
2	3	9
2	4	11
2	5	13
2	6	7
2	7	23
2	8	14
6	1	7
6	2	7
6	3	13
6	4	11
6	5	6
6	7	18
6	8	7

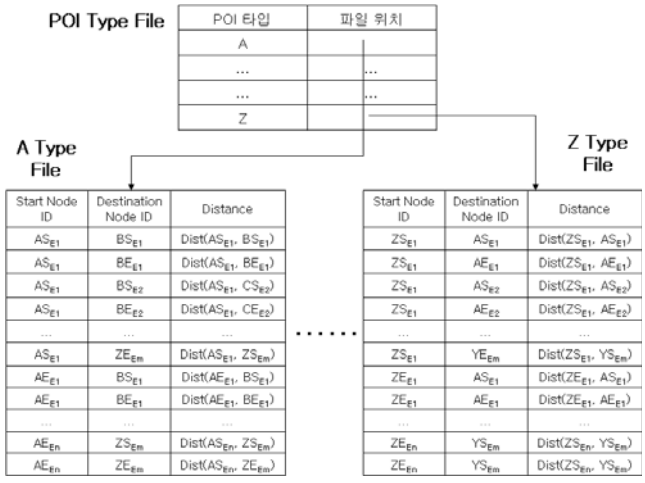
S.. Node	D.. Node	Distance
3	1	11
3	2	9
3	4	2
3	5	7
3	6	13
3	7	14
3	8	16
7	1	25
7	2	23
7	3	14
7	4	12
7	5	17
7	6	18
7	8	11

S.. Node	D.. Node	Distance
4	1	13
4	2	11
4	3	2
4	5	5
4	6	11
4	7	12
4	8	14
8	1	14
8	2	14
8	3	16
8	4	14
8	5	9
8	6	7
8	7	11

<그림 4> 노드-노드 실체화 파일의 구성

3.2 POI 기반 실체화 파일

본 논문에서는 효율적인 저장 공간의 사용 및 질의 수행에 필요한 최소한의 실체화 파일을 이용하기 위하여, POI 정보를 기반으로 하여 각 노드 쌍의 실제 거리만을 저장하는 실체화 파일 기법을 제안한다. POI 정보 기반의 실체화 파일은 주어진 POI 타입을 포함하고 있는 에지를 구성하고 있는 양 노드에서부터 다른 POI 타입을 포함하고 있는 에지의 양 노드까지의 거리만을 저장한다. 이를 이용하여 불필요한 저장 공간을 제거할 뿐만 아니라 빠른 응답 시간을 제공한다. 한편 POI 기반 실체화 파일은 기준이 되는 POI 타입을 고려하여 각 POI 타입에 대하여 실체화 파일을 생성할 수 있다. <그림 5>는 POI 타입을 고려한 실체화 파일의 레코드 구성을 나타낸다. POI 타입 파일은 기준 POI 타입에 대한 노드에서 다른 POI 타입을 가진 노드까지의 거리들을 저장하고 있는 실체화 파일의 위치를 가리키고 있다. POI 타입이 {A, B, ... , Y, Z}이고, 이 중에서 A POI 타입을 포함하고 있는 에지의 양 노드를 $\{AS_{Ei}, AE_{Ei}\}$라고 표시하고 A POI 타입의 전체 노드 개수를 n라고 한다면 전체 집합은 $\{AS_{E1}, AE_{E1}, AS_{E2}, AE_{E2}, \ldots, AS_{En}, AE_{En}\}$가 된다. 아울러 각각의 실체화 파일은 Start Node ID는 기준 POI 타입을 갖고 있는 노드, Destination Node ID는 다른 POI 타입을 갖고 있는 노드, Distance는 두 노드의 거리를 나타낸다.



<그림 5> POI 기반 실체화 파일의 구조

POI Type	File Location
R	
S	
T	

(a) R Type File

S_ Node	D_ Node	Distance
1	4	13
1	5	13
1	7	25
1	8	14
2	4	11
2	5	13
2	7	23
2	8	14

(b) S Type File

S_ Node	D_ Node	Distance
4	1	13
4	2	11
4	7	12
5	1	13
5	2	13
5	4	5
5	7	17
8	1	14
8	2	14
8	4	14
8	7	11

(c) T Type File

S_ Node	D_ Node	Distance
4	1	13
4	2	11
4	5	5
4	8	14
7	1	25
7	2	23
7	4	12
7	5	17
7	8	11

<그림 6> POI 기반 실체화 파일의 예

<그림 3>의 예제에 대해 POI 기반 실체화 파일을 구성하면 <그림 6>과 같다. 두 개의 POI 타입을 입력받아 질의를 수행할 때, <그림 6>의 실체화 파일 중에서, 개수가 적은 타입에 해당하는 파일을 이용하여 쉽게 질의를 수행할 수 있다. 제안하는 POI 기반 실체화 파일을 사용함으로써 다음과 같은 장점이 존재한다. 첫째, 노드 식별자만을 가지고 노드와 노드 사이의 최소 거리를 경로 탐색 알고리즘을 사용하지 않고 정의 1식을 이용하여 쉽게 계산할 수 있다. 둘째, POI 타입을 고려함으로써 POI를 포함하지 않는 노드는 사용할 필요가 없기 때문에, 전체 실체화 파일의 크기를 줄일 수 있다. 셋째, 해당되는 POI 타입의 실체화 파일은 질의 수행에 필요한 모든 노드 정보를 포함하고 있으므로, 최소한의 거리 정보만을 이용하여 질의를 수행할 수 있다.

<그림 7>은 제시된 POI 기반 실체화 파일을 생성하는 알고리즘이다. 첫째, 입력 받는 POI 타입에 대하여 해당 POI를 포함하고 있는 에지의 양쪽 노드를 검색한다. 둘째, 양쪽 노드에 대하여 다른 POI를 포함하고 있는 모든 노드에 대한 거리정보를 탐색한다. 셋째, 양쪽 노드에 대하여 다른 노드까지의 거리정보를 실체화 파일에 저장한다. 위와 같은 단계를 입력 받은 POI 타입에 대하여 모두

```
01.  create_POI_materialization_file(POI_type)
02.  for each (all POI_type) {
03.      node[ni, nj] = fetch_node(a POI);
04.      for each (ni and nj) {
05.          node_info_ni = all_node_destination(node ni);
06.          node_info_nj = all_node_destination(node nj);
07.          node_info = node_info_ni ∪node_info_nj
08.      }
09.      fwrite(POI_materialization_file, node_info);
10.      //info <start node id, destination node id, distance>;
11.      fwrite(POI_type_File, location); // POI_materialization_file location
12.      }
13.  }
```

<그림 7> POI 기반 실체화 파일 생성 알고리즘

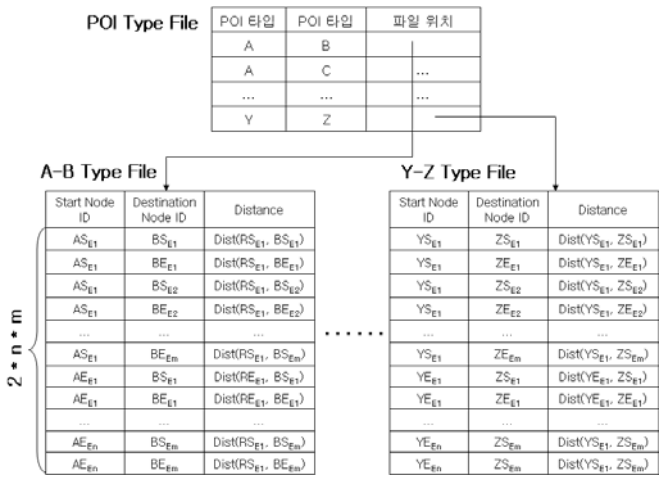
수행하면, POI를 기반 실체화 파일이 생성된다. 아울러 POI의 삽입 및 삭제가 일어날 경우, 해당 POI를 포함하고 있는 에지 또는 이웃하는 에지가 다른 POI를 포함하고 있다면 실체화 파일에는 아무런 영향을 주지 않으며, 그렇지 않은 경우 해당 노드에 대한 실체화 파일의 삽입, 삭제를 수행한다.

3.3 POI 쌍 기반 실체화 파일

만약 POI 타입의 개수가 매우 적은 경우에는 (예를 들면, 10 미만), POI 쌍을 기반으로 하여 보다 효율적인 실체화 파일을 구성할 수 있다. POI 쌍을 기반으로 한 실체화 파일은 질의 수행에 필요한 두 개의 POI 타입을 고려한 방법이다. 즉, 두 개의 POI 타입을 고려한 실체화 파일을 구성함으로써 질의 수행에 필요한 최소 노드들만을 저장한다. 따라서 앞서 제안한 POI 기반 실체화 파일보다 더욱 빠른 응답속도를 제공할 수 있다. <그림 8>은 POI 쌍 기반의 실체화 파일의 레코드 구성을 나타낸다. POI 타입 파일은 두 개의 POI 타입에 대한 실체화 파일의 위치를 가리키고 있다. 각각의 실체화 파일의 레코드 크기는 최대 두 POI 집합을 구성하는 POI 개수의 곱의 두 배 크기이다. 아울러 POI 쌍 기반 실체화 파일의 개수는 POI 타입의 개수를 N이라 한다면 ${}_{N}C_{2}$이다. 따라서 POI 타입의 개수가 매우 적은 경우가 아니라면 파일의 개수가 너무 많아지게 되어 비효율적이다.

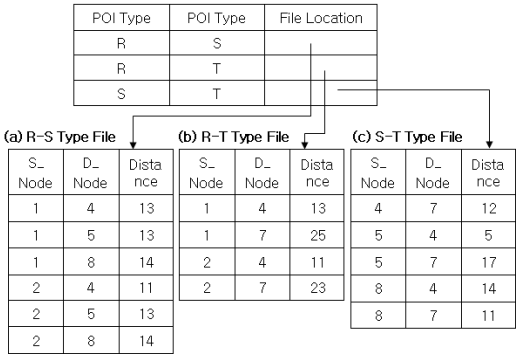
<그림 3>의 예제에 대해 POI 쌍 기반 실체화 파일을 구성하면 <그림 9>와 같다. 두 개의 POI 타입을 입력받아 질의를 수행하면, 해당 POI 쌍 실체화 파일을 이용하여 쉽게 질의를 수행할 수 있다. 제시한 POI 쌍 기반 실체화 파일을 사용함으로써 다음과 같은 장, 단점이 존재한다. 첫째, 입력 받은 두 개의 POI 타입을 고려한 실체화 파일은 질의 수행에 필요한 모든 정보를 포함하고 있으므로 빠른 질의응답 시간을 제공할 수 있다. 둘째, 다른 POI 타입을 고려한 정보가 없으므로 읽어야 할 파일의 크기가 필요한 정보의 크기와 같다. 반면 POI 타입의 개수가 크다면 생성하여야 할 실체화 파일의 개수가 많아지므로 POI 타입의 개수가 매우 적을 때 효율적인 방법이다.

<그림 10>은 제시된 POI 쌍 기반 실체화 파일을 생성하는 알고리즘이다. 첫째, 입력 받는 POI 타입에 대하여 해당 POI를 포함하고 있는 에지의 양쪽 노드를 검색한다. 둘째, 양쪽 노드에 대하여 입력 받은 다른 POI를 포함하고 있는 모든 노드에 대한 거리정보를 탐색한다. 셋째, 양쪽 노드에 대하여 다른 노드까지의 거리정보를 실체화 파일에 저



<그림 8> POI 쌍 기반 실체화 파일의 구조

장한다. 위와 같은 단계를 입력 받은 POI 타입에 대하여 수행하면 POI 쌍 기반 실체화 파일이 생성된다. 아울러 POI의 삽입 및 삭제가 일어날 경우, POI 기반 실체화 파일보다 간단히 수행된다. 이는 삽입 및 삭제가 일어난 POI의 타입에 따라 해당 실체화 파일에서만 삽입 및 삭제가 수행되기 때문이다.



POI Type	POI Type	File Location
R	S	
R	T	
S	T	

(a) R-S Type File

S_ Node	D_ Node	Distance
1	4	13
1	5	13
1	8	14
2	4	11
2	5	13
2	8	14

(b) R-T Type File

S_ Node	D_ Node	Distance
1	4	13
1	7	25
2	4	11
2	7	23

(c) S-T Type File

S_ Node	D_ Node	Distance
4	7	12
5	4	5
5	7	17
8	4	14
8	7	11

<그림 9> POI 쌍 기반 실체화 파일의 예

4. POI 정보 기반의 실체화 기법을 이용한 질의처리 알고리즘

효율적인 Closest Pairs 질의처리 알고리즘 및 e-distance 조인 질의처리 알고리즘을 제안하기 위하여, 본 논문에서는 3장에서 제시한 두 가지 실체화 파일 구성 방법 가운데, POI 타입 개수에 민감하지 않은 보다 범용적인 방법인 POI 기반 실체화 파일을 사용한 알고리즘을 제안한다.

4.1 Closest Pairs 질의처리 알고리즘

본 논문에서 제안하는 POI 기반 실체화 파일을 이용한 Closest Pairs 질의 처리 알고리즘은, R-Tree 를 이용한 조인(join)을 통해 후보 POI 쌍을 탐색하여 유클리디언 거리가 작은 후보 POI 쌍들을 구하고, 이들의 실제 거리 계산을 위해 POI 기반 실체화 파일을 사용하는 방법이다. 이는 실제 거리를 계산하는데 있어, 질의점을 포함하고 있는 에지와 POI를 포함하고 있는 에지를 바로 알 수 있기 때문에, 네트워크 거리 계산 알고리즘에 근거하여 실제거리를 쉽게 계산할 수 있는 장점이 있다. 아울러 R-Tree 조인을 통하여 전체 POI쌍들 중에서 거리가 가까운 POI 쌍을 후보 셋으로 선정함으로써, 전체 POI 쌍을 계산할 필요가 없다. 제안하는 Closest Pairs 질의처리 알고리즘은 <그림 11>과 같다. 먼저, 기준이 되는 POI 타입에 대하여 실체화 파일을 메모리에 상주시킨다. 둘째, R-tree 조인에 의한 후보 POI 쌍을 구하여 거리에 따라 정렬한다. 셋째, 정렬한 후보 POI 쌍들에 대하여 POI 기반 실체화 파일을 이용하여 실제 거리를 계산한다. 이 때, k개의 후보 POI 쌍을 검색하였다면 상계값(ndmax)을 설정한다. 상계값은 알

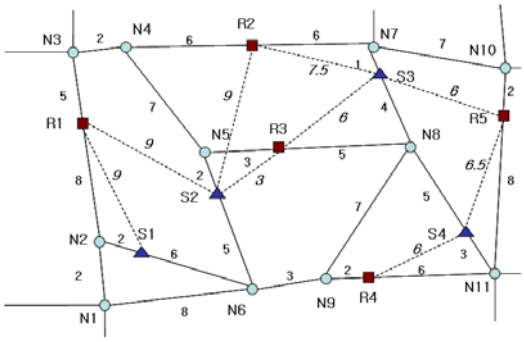
```
01.  create_POI_materialization_file(R POI_type, S POI_type)
02.  for each (R POI_type) {
03.       node[ni, nj] = fetch_node(a POI);
04.       for each (ni and nj) {
05.            node_info_ni = all_node_destination(node ni, S POI_type);
06.            node_info_nj = all_node_destination(node nj, S POI_type);
07.            node_info = node_info_ni ∪node_info_nj
08.       }
09.       fwrite(POI_materialization_file, node_info);
10.       //info <start node id, destination node id, distance>;
11.       fwrite(POI_type_File, location); // POI_materialization_file location
12.       }
13.  }
```

<그림 10> POI 쌍 기반의 실체화 파일 생성 알고리즘

```
PMCP(R, S, k)      // POI based Materialization Closest Pairs Algorithm
// R, S : two entity data sets, k : the number of threshold,
// the number of R < the number of S
load_mater_file(R type);
Rjoin_result = R-tree-join(R, S);
sort Rjoin_result on R;
i = 0, ndmax = ∞;
for each node r ∈ R in Rjoin_result {
      <n1, n2> = find_edge(r);
      for each POI s ∈ S {
              if (ndmax < Euclidean_dist(s,t)) return result;
              <n3, n4> = find_edge(s);
              distance = calculate_distance(n1, n2, n3, n4);
              if (i < k) result = result ∪ (s,t);
              else if (distance < ndmax) {
                      result = result ∪ (s,t);
                      update(ndmax);
              }
      } //for each POI
}// for each node
```

<그림 11> 제안하는 Closest Pairs 질의처리 알고리즘

고리즘 수행을 종료하기 위한 값으로써, 다음 후보 POI 쌍의 유클리디언 거리가 상계값보다 크다면 더 이상 알고리즘을 수행할 필요가 없으므로 결과 셋을 반환한다. 만약, 후보 POI 쌍의 실제 거리가 상계값보다 작다면 상계값을 갱신한다.



<그림 12> 제안하는 Closest Pairs 질의처리 알고리즘 예

<그림 12>는 제안하는 Closest Pairs 질의처리 알고리즘을 이용하여 "실제 거리가 가까운 3개의 POI 쌍을 검색"하는 예제이다. 첫째, R-Tree 조인을 통하여 R 셋과 S 셋을 조인한다. 조인된 결과의 POI 쌍을 유클리디언 거리로써 정렬하여 후보 POI 셋을 구한다. 정렬된 후보 POI 셋은 <(R3, S2, 3), (R4, S4, 6), (R3, S3, 6), (R3, S5, 6), (R5, S4, 6.5), ... >이다. 둘째, 메모리에 상주된 실체화 파일에서 R3가 포함되어 있는 노드 N5 및 노드 N8에 대해, S2가 포함되어 있는 노드 N5 및 노드 N6까지의 거리를 이용하여 POI 쌍 사이의 실제 거리를 계산한다. 이와 같이 3개의 POI 쌍을 찾고 결과 셋에 <(R3, S2, 5), (R4, S4, 9), (R3, S3, 9)>를 삽입한 후에 상계값을 9로 설정한다. 셋째, 다음 후보 POI 쌍의 유클리디언 거리가 6이므로, 상계값보다 작다. 따라서 이에 대한 계산을 수행하여 실제 거리 10을 구한다. 이는 상계값보다 크므로 결과셋에 넣지 않는다. 다음 후보 POI 쌍이 유클리디언 거리가 상계값보다 클 때까지 과정을 반복한다. 최종 결과 집합은 {(R3, S2, 5), (R2, S3, 7), (R4, S4, 9)}가 된다. 상계값이 9로 설정

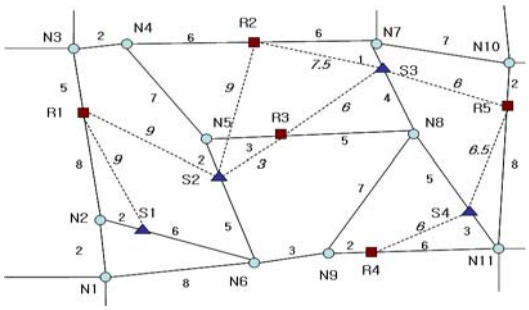
```
PMJ(R, S, e)      // POI based Materialization e-distance Join Algorithm
// R, S : two entity data sets, e : range,
// the number of R < the number of S
load_mater_file(R type);
Rjoin_result = R-tree-join(R, S);
Rjoin_result = cut_result(Rjoin_result, e);
sort Rjoin_result on R;
i = 0, ndmax = ∞;
for each node r ∈ R in Rjoin_result {
      <n1, n2> = find_edge(r);
      for each POI s ∈ S {
            <n3, n4> = find_edge(s);
            distance = calculate_distance(n1, n2, n3, n4);
            if (distance < ndmax) {
                  result = result ∪ (s,t);
            }
      } // for each POI
} // for each node
```

<그림 13> 제안하는 e-distance 조인 질의처리 알고리즘

되었기 때문에 실제 계산에 필요한 후보 POI 쌍은 총 5쌍이다. 따라서 제안하는 알고리즘은 실제 POI까지의 거리를 구하는 쌍의 개수를 크게 감소시킬 수 있다. 이는 CPU 비용뿐만 아니라 디스크 I/O 횟수까지 줄일 수 있으므로 매우 효과적이다.

4.2 e-distance 조인 질의처리 알고리즘

e-distance 조인 질의처리 알고리즘은 Closest Pairs 질의처리 알고리즘과 같이 POI 기반 실체화 파일을 사용하여 디스크 I/O 및 검색에 필요한 노드간의 거리계산을 줄임으로써 성능을 향상시킬 수 있다. 본 논문에서 제안하는 e-distance 조인 질의처리 알고리즘은 <그림 13>과 같다. 첫째, 해당 POI 타입의 노드-노드 거리를 저장하고 있는 POI 기반 실체화 파일을 메모리에 상주시킨다. 둘째, R-tree 조인에 의한 후보 POI 쌍을 구하여 거리에 따라 정렬한다. 이 때, 유클리디언 거리가 범위 e이내인 후보 POI 쌍만을 정렬하여 구한다. 이는 실제 POI 쌍 사이의 거리가 유클리디언 거리보다 작을 수 없기 때문에 후보 셋을 쉽게 구할 수 있다. 셋째, 정렬한 후보 POI 쌍들에 대하여 POI 기반 실체화 파일을 사용하여 실제 거리를 계산한다. 계산된 실제 거리가 범위 e보다 작다면 결과 집합에 삽입한다. 이 과정을 모든 후보 POI 쌍에 대하여 수행한 후, 알고리즘은 종료된다.



<그림 14> 제안하는 e-distance 조인 질의처리 알고리즘 예

<그림 14>는 제안하는 e-distance 조인 질의처리 알고리즘을 이용하여 "실제 거리가 범위 8 이내인 POI 쌍을 검색"하는 예제이다. 첫째, R-Tree 조인을 통하여 R 셋과 S 셋을 조인한 POI 쌍을 유

클리디언 거리로써 정렬하여 후보 POI 셋을 구한다. 이 때, 범위 8 이내인 정렬된 후보 POI 셋은 <(R3, S2, 3), (R4, S4, 6), (R3, S3, 6), (R3, S5, 6), (R5, S4, 6.5), (R2, S3, 7.5)>이다. 둘째, 정렬된 후보 POI 셋에 대하여 메모리에 상주된 POI 기반 실체화 파일을 이용하여 실제 거리를 계산한다. 예를 들면, R3와 S2 쌍에서 R3가 포함되어 있는 노드 N5 및 노드 N8에 대해, S2가 포함되어 있는 노드 N5 및 노드 N6까지의 거리를 이용하여 POI 쌍 사이의 실제 거리를 계산한다. 이와 같이 실제 거리를 계산하여 범위 8이내인 POI 쌍을 찾으면, 최종 결과 집합은 <(R3, S2, 5), (R2, S3, 7)>가 된다. 예제에서 실제 거리계산에 필요한 후보 POI 쌍은 총 6쌍이다. 따라서 제안하는 e-distance 조인 질의처리 알고리즘은 후보 POI를 쉽게 구할 수 있을 뿐만 아니라, 실제 POI간의 거리를 쉽게 구할 수 있기 때문에 전체 응답 시간을 크게 감소시킬 수 있다.

5. 성능 평가

제안하는 Closest Pairs 질의처리 알고리즘 및 e-distance 조인 질의처리 알고리즘이 기존 알고리즘에 비해 검색 성능이 우수함을 보이기 위해 성능평가를 수행한다. 성능 평가 환경은 <표 1>과 같다. 지도 데이터는 실제 샌프란시스코 만 지도를 사용하였고 [11], 전체 노드 수는 175,343개, 전체 에지 수는 223,199개로 구성되어 있다. 아울러, RunTime21[12] 알고리즘을 사용하여 임의로 생성한 여러 타입을 가진 10846개의 POI를 사용하였고, POI 기반 실체화 파일의 전체 크기는 약 1.7GB이다. 아울러 질의 성능평가 및 알고리즘 구현을 위해 이미 개발된 공간 네트워크를 위한 저장/색인 구조를 사용하였다[13]. 성능비교를 위한 기존 알고리즘은 HKUST 연구에서 제안된 대표적인 Closest Pairs 질의처리 알고리즘인 CPER, CPNE 알고리즘과, 아울러 대표적인 e-distance 조인 질의처리 알고리즘인 JER, JNE 알고리즘을 대상으로 한다.

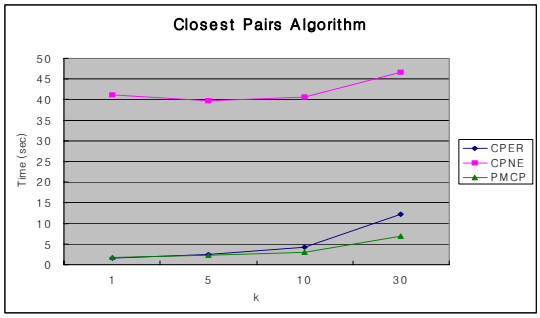
<표 1> 실험 환경

Memory	3 GB
CPU	3.0 GHz Dual
OS	Windows Server 2003 Enterprise
Tool	Visual C++ 7.1

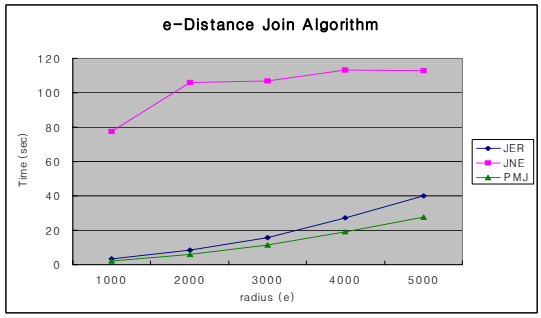
<표 2>는 CPER, CPNE, 제안하는 Closest Pairs 질의처리 알고리즘(POI-based Materialization-oriented Closest Pairs Algorithm, 이하 PMCP 라 명명)의 전체 탐색시간 결과를 나타낸다. 성능측정은 k의 값을 1부터 100까지 변화시키면서 전체 응답시간(Wall Time)을 측정한 것이다. <표 2> 에서 나타난 것과 같이, k=1일 때 전체 응답 시간은 CPER의 경우 1.6초, CPNE의 경우 약 41초, PMCP는 1.6초로 나타난다. 제안하는 방법인 PMCP 은 CPNE보다는 월등한 성능을 보이며 CPER과는 비슷한 성능을 나타낸다. 한편 k=10일 때, CPER의 전체 응답시간은 4.1초, CPNE는 40초, PMCP는 3.1초로 나타난다. 마지막으로 k 가 30일 경우에는, CPER은 12.2초, CPNE는 46초, PMCP는 6.9초로써, 제안하는 알고리즘인 PMCP가 기존 연구인 CPER과 CPNE에 비해서 월등히 효율적임을 알 수 있다. <그림 15>는 CPER, CPNE, PMCP의 전체 응답시간을 k값에 따라서 그래프로 표시한 것이다. CPER과 PMCP는 k값이 증가할수록 전체 응답시간이 선형적인 증가를 보이고 있는 반면, CPNE는 k값의 증가에 비해 일정한 응답시간을 보이고 있으나 CPER과 PMCP에 비해 성능이 매우 저하됨을 알 수 있다. 아울러 k값이 5 이상에서는, 제안하는 PMCP가 기존 CPER 및 CPNE에 비해서 성능이 우수함을 알 수 있다.

<표 2> Closest Pairs 질의 성능비교 (단위 : sec)

k	CPER	CPNE	PMCP
1	1.672707	41.14862	1.689485
5	2.479201	39.80097	2.289634
10	4.193967	40.61347	3.062442
30	12.22203	46.57938	6.908324



<그림 15> Closest Pairs 질의 성능비교



<그림 16> e-distance 조인 질의 성능비교

<표 3>은 기존 JER, JNE, 제안하는 e-distance 조인 질의처리 알고리즘(POI-based Materialization-oriented e-distance Join Algorithm: PMJ)의 전체 탐색시간 결과를 나타낸다. 범위(radius)는 실제 데이터의 거리를 의미하며 단위는 미터(m)이다. <표 3>에 나타난바와 같이, 범위가 500m 일 때, JER 은 40초, JNE 는 112초, 제안하는 PMJ 는 약 27 초이다. 이를 통해 JNE 는 JER 및 PMJ 에 비해 성능이 월등히 저하됨을 알 수 있고, JER 과 PMJ 사이의 성능비교에서는 본 논문에서 제안하는 PMJ 알고리즘이 JER 기법에 비해 약 50% 이상의 성능 향상을 나타냄을 알 수 있다. <그림 16> 은 JER, JNE, PMJ의 전체 응답시간을 범위에 따른 그래프로 표시한 것이다. 그래프에서 알 수 있듯이, 범위가 증가할수록 JER에 비해 PMJ 가 보다 성능이 우수함을 알 수 있다. 이는 POI 기반 실체화 파일에 저장된 정보를 사용함으로써 기존 알고리즘에서 필요한 노드 접근을 위한 디스크 I/O 및 거리 계산에 필요한 CPU 시간을 줄일 수 있기 때문이다.

<표 3> e-distance 조인 질의 전체 검색 시간

(단위 : sec)

radius (m)	JER	JNE	PMJ
100	3.31872	77.4427	2.14760
200	8.46558	105.8177	5.83021
300	15.58430	106.9423	11.34844
400	27.13744	113.0050	19.08724
500	40.06246	112.8493	27.71948

6. 결 론

본 논문에서는 공간 네트워크 데이터베이스를 위한 기존 알고리즘의 문제점인 디스크 I/O 횟수와 거리 계산으로 인한 성능감소를 극복하기 위해, POI 기반 실체화 파일을 이용하여 보다 효율적인 Closest Pairs 질의 및 e-distance 조인 질의처리 알고리즘을 제안하였다. 제안한 Closest Pairs 질의처리 알고리즘은 기존 CPER 알고리즘보다 k가 커짐에 따라 최고 약 2배, CPNE 알고리즘보다는 최고 약 25 배의 성능이 우수함을 알 수 있다. 또한, 제안한 e-distance 조인 질의처리 알고리즘은 범위가 커짐에 따라 기존 JER보다 약 1.5배, JNE 보다 최고 약 30배의 성능이 우수함을 입증하였다. 한편 본 논문에서는 제안하는 Closest Pairs 질의 및 e-distance 조인 질의처리 알고리즘은 범용성 문제 때문에 POI 기반 실체화 파일을 이용하여 설계되었다.

향후 연구로는 제시한 알고리즘은 단일 시스템에서 적용이 가능하도록 설계 되었으므로, 공간 네트워크 영역을 지역적으로 분산 영역으로 분할하여 저장함과 더불어 이를 효율적으로 처리할 수 있는 분산 시스템 환경에서의 질의처리 알고리즘을 개발하는 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. 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.
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, 2003, pp. 255-273.
5. M. Kolahdouzan and C. Shahabi, "Voronoi-Based K Nearest Neighbor Search for Spatial Network Databases", Proc. of VLDB, 2004, pp. 840-851.
6. D. Papadias, J. Zhang, N. Mamoulis, and Y. Tao, "Query Processing in Spatial Network Databases" Proc. of VLDB, 2003, pp, 802-813.
7. Z. Song, and N. Roussopoulos, "K-Nearest neighbor Search for Moving Query Point", Proc. of SSTD, 2001, pp. 79-96.
8. C.S. Jensen, T.B Pedersen, L. Speicys, and I. Timko, "Data Modeling for Mobile Services in the Real World", Proc. of SSTD, 2003, pp. 1-9.
9. S. Shekhar, and D.R. Liu, "CCAM: A Connectivity-Clustered Access Method for Networks and Network Computations", Proc. of IEEE Tran. on Knowledge and Data Engineering, Vol. 9, No. 1, 1997, pp. 102-119.
10. 김용기, 니하드 카림 초우더리, 이현조, 장재우, "공간 네트워크 데이터베이스에서 실체화 기법을 이용한 범위 및 k-최근접 질의처리 알고리즘", 한국공간시스템학회 논문지 Vol. 9, No. 2, 2007, pp. 67-79.
11. http://www.fh-oow.de/institute/iapg/personen/brinkhoff/generator/
12. T. Brinkhoff, "A Framework for Generation Network - Based Moving Objects", Geo-Informat.
13. 강홍민, 장재우, "공간 네트워크 데이터베이스를 위한 저장 및 색인 구조의 설계", 한국정보과학회 가을 학술발표논문집, 제31권, 제2호, 2004, pp.133-136.

김용기
2002년 전북대학교 컴퓨터공학과(공학사)
2005년 전북대학교 대학원 컴퓨터공학과(공학석사)
2006년~현재 전북대학교 대학원 컴퓨터공학과 박사과정
관심분야 : 공간 데이터베이스, 질의처리 알고리즘, 공간 색인 구조

장재우
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, 상황인식, 하부저장구조