다차원 공간에서 거리조인 질의처리를 위한 R-트리의 효율적 접근

(Efficient Accesses of R-Trees for Distance Join Query Processing in Multi-Dimensional Space)

신 효 섭 † 문 봉 기 †† 이 석 호 †††
(Hyoseop Shin) (Boongki Moon) (Sukho Lee)

요 약 거리조인은 R-트리를 사용하여 두 공간 데이타 집합 사이의 데이타쌍을 거리 상 가까운 순으로 검색하는 공간조인이다. 거리조인은 R-트리를 하향식으로 순회하면서 생성되는 노드쌍들을 거리값 순으로 우선순위 큐에 저장한다. 본 논문에서는 거리조인 처리시 우선순위 큐 안에서 동점자 노드쌍들의 우선 순위 정책이 알고리즘의 성능을 많이 좌우할 수 있음을 보여주고, 이를 위한 최적화된 2차 우선 순위 기법을 제안한다. 실험을 통하여, 제안한 기법이 다른 기법에 비하여 항상 좋은 성능을 나타냄을 보여준다.

키워드 : 거리 조인, R-트리, 2차 우선 순위

Abstract The distance join is a spatial join which finds data pairs in the order of distance between two spatial data sets using R-trees. The distance join stores node pairs in a priority queue, which are retrieved while traversing R-trees in a top-down manner, in the order of distance. This paper first shows that a priority strategy for the tied pairs in the priority queue during distance join processing has much effect on its performance, and then proposes an optimized secondary priority method. The experiments show that the proposed method is always better than the other methods in the performance perspectives.

Key words : Distance Join, R-Tree, Secondary Priority

1. 서 론

거리조인(distance join)은 2차원 이상의 다차원 공간 상의 두 데이타 집합에 대하여 공간적인 거리에 기반하여 거리가 가까운 순으로 데이타 쌍을 검색하는 조인 연산이다[1,2,3]. 거리조인 질의의 예를 들면 다음과 같다.

```
Select a, b
From A a, B b
Order By distance(a, b)
Stop After K;
```

위의 SQL 질의문에서 Order By 문은 질의 결과를 공간적인 거리의 오름차순으로 정렬할 것을, 그리고 Stop After 문은 질의 결과 카디날리티를 K로 제한함을 명시하는 구문이다. 즉, "두 데이타 집합 A, B 에 대하여 서로 거리가 가장 가까운 K 쌍을 검색하여 거리값이 작은 순으로 나열하라"는 뜻으로 해석된다.

이와 같은 거리조인 질의의 응용은 공간 데이타베이스 시스템, 지리정보 시스템, 이미지 데이타베이스 시스템, 항공기 운항 관제 시스템, CAM/CAM 등 데이타를 다차원 공간 상에서 점(point) 이나 다면체(hyper- polygon)로 표현하여 이용하는 여러 분야에서 찾아볼 수 있다. 예를 들어, 공간 데이타베이스 시스템에서는 "서울 시내 안에서 지리적으로 가까이 위치한 호텔과 레스토랑을 검색하라"와 같이 공간적으로 가까이 위치한 공간 객체쌍을 검색하는 응용이 있을 수 있다. 이미지 검색 시스템에서는 서로 다른 두 이미지 데이타 집합을 대상으로 모양이나 색깔이 가장 유사한 이미지 쌍을 찾는 응용에 적용할 수 있다.

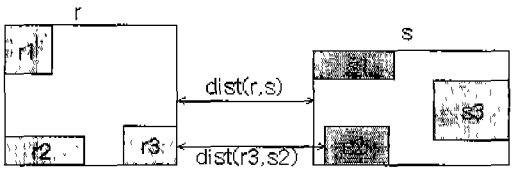
1.1 R-트리를 이용한 거리조인의 개요

거리조인은 대상이 되는 두 데이타 집합이 R-트리와 같은 공간 인덱스[5,6,7,8,9,10]로 구성되었을 때 효율적

† 비 회 원 : 서울대학교 컴퓨터공학부
hsshin@db.snu.ac.kr
†† 비 회 원 : 미국 아리조나대학교 전산학과 교수
bkmoon@cs.arizona.edu
††† 종신회원 : 서울대학교 컴퓨터공학부 교수
shlee@csc.snu.ac.kr
논문접수 : 2000년 12월 18일
심사완료 : 2001년 6월 26일

으로 수행된다. 두 R-트리를 하향식(top-down)으로 순회하면서 짝을 지으면 탐색 공간을 그만큼 줄일 수 있기 때문이다[1,2,3,4]. R-트리를 하향식으로 순회하는 동안에 검사한 노드쌍을 우선순위 큐에 저장하는 것이 바람직하다. 큐의 우선 순위는 노드쌍 안에서 두 노드 간의 **거리**(distance)이다. 이렇게 함으로써 두 노드 간의 거리값이 작은 쌍부터 순회하게 한다. 이 우선 순위 큐를 **메인큐**(main queue)라고 정의한다.

메인큐는 양쪽 트리의 루트 노드쌍으로 초기화된다. 메인큐로부터 객체가 아닌 노드가 포함되어 있는 비객체노드 쌍이 삭제되어 검출되면 노드쌍의 한쪽 노드의 자식노드들이 나머지 다른 쪽 노드의 자식노드들과 짝을 지어서 메인 큐에 재삽입된다. 이 **노드 확장**(node expansion) 과정은 메인 큐가 비거나, 또는 질의 결과가 모두 도출되면 종료된다. 만약 메인큐로부터 검출된 노드쌍이 객체들로만 구성된 객체노드 쌍이라면, 그 노드쌍은 질의 결과로서 즉시 반환된다. 예를 들어, (그림 1)의 (a)에서 두 R-트리 노드쌍 <r,s>가 메인큐에서 삭제되어, 노드 확장을 하면 {<r1,s1>, <r1,s2>, <r1,s3>, <r2,s1>, <r2,s2>, <r2,s3>, <r3,s1>, <r3,s2>, <r3,s3>} 등의 9쌍의 노드쌍들을 생성하며, 이들은 메인큐에 재삽입된다. 이들 재삽입된 자식 노드쌍들의 거리는 원래의 부모노드쌍인 <r,s>의 거리보다 항상 크거나 같다는 성질을 가진다.



(a) 다차원 객체의 계층적 구성

(b) 다차원 객체의 R-트리 구성

그림 1 R-트리의 공간 계층성

거리조인을 수행하기 이전에 검색한 객체 노드쌍의 개수 K가 미리 정해져 있다면 이를 이용하여 알고리즘의 성능 향상을 기대할 수 있다. 알고리즘 수행 단계에서 현재까지의 객체 노드쌍의 거리 중 가장 작은 K개를 유지하고, 그 중 가장 큰 거리값을 한계값 **qDmax**라고 정하고, 차후 생성되는 노드쌍 중 거리가 이 값보다 더 큰 값을 가지는 것들은 메인큐에 삽입하지 않고 제거할 수 있다. K개의 최소 거리값을 유지하기 위하여 메인큐와는 별도의 **거리큐**(distance queue)를 둔다. 거리큐는 K길이의 최대히프(max heap)로 구성할 수 있다. qDmax는 거리큐의 헤더값을 나타내는데 알고리즘 초기에는 무한대값으로 지정되며, 알고리즘이 수행됨에 따라 점점 더 작아지며, K에 대한 실제 한계거리값 **Dmax**보다는 항상 크거나 같게 된다. 하지만 Dmax값은 알고리즘이 끝난 이후에나 알 수 있는 값이다. 거리큐를 이용한 거리조인을 **K 거리조인(KDJ, K Distance Join)**이라고 정의한다[1]. (그림 2)는 K 거리조인의 기본 구조를 표현한 것이다.

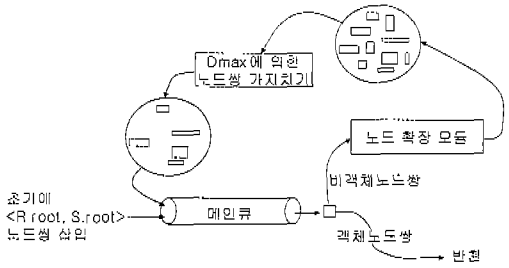


그림 2 K 거리조인의 기본 구조[1]

1.2 문제 정의

K 거리조인에서는 노드확장에서 생성된 노드쌍들 중 거리값이 qDmax보다 큰 것들은 가지치기하고 남은 것들만 메인큐에 삽입된다. qDmax 값은 알고리즘 초기에는 무한대값으로 설정되고 알고리즘이 진행됨에 따라 그 값이 작아진다. qDmax값은 객체 노드쌍들의 거리값들이 거리큐(DQ)에 삽입되면서 갱신되므로, 작은 값의 거리값이 알고리즘 초기에 많이 생성될수록 qDmax값의 감소 속도가 증가한다. 이 qDmax를 감소시킬 수 있는 객체 노드쌍들은 메인큐로부터 나온 비객체 노드쌍이 노드확장 모듈을 거치면서 생성된다. 그런데 이 비객체 노드쌍들은 두 R-트리의 중간 노드들끼리의 조합이므로, 대개의 경우 공간적으로 서로 겹쳐진다. 즉, 메인큐에서 삭제되는 중간노드쌍은 거리가 0인 경우가 대부분이다. 이들은 메인큐 안에서 거리가 0으로 동일하기 때문에 이들간에는 아무런 우선 순위가 없다고 볼 수 있다. 실험에 따르면 알고리즘 수행 도중 메인큐로부터 삭제된 비객체 노드쌍의 98% 이상이 거리가 0이었다. 따라서 이렇게 겹쳐져 있는 노드쌍들 간의 메인큐에서

의 우선 순위는 노드 확장에 의해서 생성되는 자식 노드쌍들이 생성되는 순서에 차이를 주게 되고, 결국 qDmax 값의 감소 속도에 많은 영향을 미칠 수 있다. 그러므로 메인큐의 우선 순위를 노드쌍들의 거리로만 정하는 것은 알고리즘의 성능 상 충분치 못하며, 메인큐의 거리값이 0인 동점자 노드쌍 처리를 위한 2차 우선 순위의 효율적인 정책이 필요하다.

이에 본 논문에서는 거리 조인 알고리즘 수행시 R-트리 노드쌍들의 최적화된 접근 우선 순위를 제안하고자 한다. 본 논문의 구성은 다음과 같다. 2절에서는 관련 연구를 기술한다. 3절에서는 R-트리 노드의 효율적인 접근을 위한 메인큐의 2차 우선 순위로서 유력한 후보 2가지를 소개한다. 4절에서는 최적화된 우선 순위를 제안한다. 5절에서는 제안된 기법에 대한 성능평가를 보인다. 마지막으로 6절에서는 본 논문의 결론을 맺는다.

2. 관련 연구

거리조인 알고리즘에 대한 연구는 [1,2,3,4] 등에서 찾아볼 수 있다. 특히 [1,2]에서는 점진적 거리 조인과 K 거리 조인을 구분하여 기술하고 있다. 본 논문에 기술하는 효율적인 R-트리 노드 접근을 위한 메인 큐 우선 순위의 최적화 기법은 주로 K 거리 조인에 적용된다.

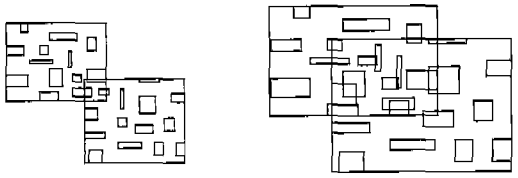
메인 큐 우선 순위에 대해서는 기존 연구에서도 간단히 언급하고 있으나, 그 목적이 명확히 정의되어 있지 않으며, 따라서 기법도 제한적이다. [3]에서는 R-트리에서 단말노드에 더 가까이 위치한 노드를 가지는 노드쌍에 더 높은 우선 순위를 주는 방식을 제안하였다(이하 깊이 우선 방식). 이것은 같은 거리값이라면 비객체 노드쌍보다는 객체 노드쌍을 먼저 생성한다는 휴리스틱에 기반하고 있다. 하지만 이 방식은 거리가 작은 노드쌍들을 되도록 빨리 생성해야 한다는 목적에 부합하지 않는다. [4]에서는 면적이 더 큰 노드를 가지는 노드쌍에 더 좋은 우선 순위를 주는 방식을 제안하였다(이하 넓이 우선 방식). 이 방식 또한 거리가 작은 노드쌍들을 되도록 빨리 생성해야 한다는 목적에 잘 부합하진 않는다. 5절 성능 평가에는 제안된 방식과 기존 방식과의 성능 비교가 포함되어 있다.

3. 메인큐의 2차 우선 순위

메인큐의 1차 우선 순위는 거리이지만 앞서 언급한 바와 같이 효율적인 R-트리 노드의 접근을 위한 메인큐의 2차 우선 순위는 qDmax값을 가능한한 빨리 감소시킬 수 있도록 정해져야 한다. 다시 말하면 거리값이 더 작은 노드쌍들이 거리값이 더 큰 노드쌍보다 더 일찍 생성되도록 정해져야 한다.

3.1 최대 거리 우선 방식

각 노드쌍의 두 노드간의 최대 거리를 따져서 그 값이 작은 노드쌍에 더 높은 우선 순위를 주는 방식이다. 노드쌍의 최대 거리가 더 작다는 것은 거리값이 더 작은 범위 안에서 자식노드쌍들이 생성됨을 의미하기 때문에 qDmax를 더 빨리 줄여줄 수 있는 가능성이 있다. 하지만 이 방식이 항상 바람직한 결과를 보여주지는 않는다. (그림 3)의 두 노드쌍 A, B를 고려해보자.



(a) 노드쌍 A (b) 노드쌍 B

그림 3 최대 거리 우선 방식의 반례

노드쌍 A의 최대거리가 노드쌍 B의 최대거리보다 작기 때문에 이 방식에 따르면 노드쌍 A가 B 보다 더 나은 우선 순위를 가지게 되지만, 실제로 노드 확장 과정에서 생성되는 노드쌍들을 고려해 볼 때 오히려 노드쌍 B가 겹침 정도가 커서 더 작은 거리값을 가지는 노드쌍이 더 많기 때문에 노드쌍 B에게 더 나은 우선 순위를 주는 게 옳다. 이처럼 최대거리 우선 방식은 노드의 겹침 정도를 반영하지 못한다는 단점이 있다.

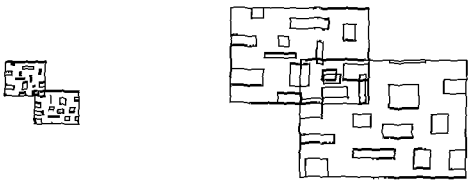
3.2 겹침 비율 우선 방식

최대 거리 우선 방식에서 발견된 문제를 보완한 방식이 겹침 비율 우선 방식이다. 두 노드의 겹침 정도가 더 큰 노드쌍에 더 높은 우선 순위를 주는 방식이다. 노드쌍의 겹침 비율은 두 노드의 겹치는 영역의 면적을 두 노드의 면적의 합으로 나눈 것으로, 다음과 같이 계산한다.

$$Overlap\_Rate(\langle R, S\rangle) = \frac{area(R \cap S)}{area(R) + area(S)} \quad (1)$$

두 노드의 겹침 비율이 크면 그만큼 거리값이 작은 노드쌍을 많이 생성할 수 있게 된다. 하지만 이 방식도 항상 바람직한 결과를 보여주진 않는다. (그림 4)의 두 노드쌍 A, B를 고려해보자.

겹침 비율값은 노드쌍 B가 노드쌍 A보다 더 크기 때문에 이 방식에 따르면 노드쌍 B가 노드쌍 A보다 더 좋은 우선 순위를 가지게 되지만, 실제로 생성되는 노드쌍들의 거리값들을 비교해 볼때는 노드쌍 A가 평균적으



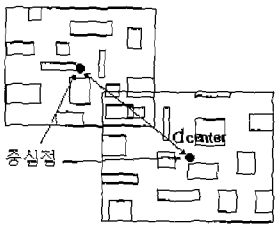
(a) 노드쌍 A (b) 노드쌍 B

그림 4 겹침 비율 우선 방식의 반례

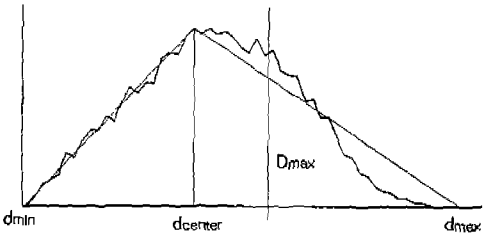
로 더 작은 거리값의 노드쌍을 더 많이 생성하게 된다. 즉, 겹침 비율 우선 방식은 두 노드의 거리 범위를 반영하지 못한다.

4. 최적의 2차 우선 순위 방식

2 절에서 나열한 2차 우선 순위 방식들은 나름의 제한점들을 가지고 있어서 어느 한 방식을 최적이라고 단정할 수 없다. 2차 우선 순위는 qDmax값을 가능한 빨리 감소시키는 것을 목표로 하는데, 이를 좀더 정확히 표현하면 qDmax값이 K에 대한 한계 거리값 Dmax에 좀더 빨리 접근할 수 있게 하기 위함이다. 그러므로 최적의 2차 우선 순위 방식은 다음과 같이 정의된다. **최적의 2차 우선 순위 방식**은 K에 대한 한계 거리값 Dmax보다 작은 거리값을 가지는 자식 노드쌍을 최대로 생성할 수 있는 노드쌍을 선택하는 방식이다.



(a) 거리가 0인 노드쌍



(b) 자식 노드쌍의 거리값 분포

그림 5 거리가 0인 노드쌍과 그에 대한 자식 노드쌍의 거리값 분포

(그림 5)의 (a)에서 두 직사각형으로 구성된 노드쌍에 대해서 자식 노드들끼리의 노드쌍의 거리값의 분포도는 (그림 5)의 (b)와 같다. x축은 자식 노드쌍의 거리를, y축은 그 거리에 해당되는 자식 노드쌍의 개수를 나타낸다. 그림에서 알 수 있듯이 거리값은 두 노드간의 최소거리(d_{min})와 최대거리(d_{max})사이에 분포한다. 그리고 두 노드의 중심점간의 거리(d_{center})에서 최대의 빈도수를 나타낸다.

이 분포도에 나타난 분포함수를 f(t)라고 할 때, 생성될 수 있는 전체 자식 노드쌍의 개수에 하여 거리값이 Dmax 안에 존재하는 노드쌍의 비율을 나타내는 확률 P(Dmax)이 다음과 이 표현된다.

$$P(Dmax) = \frac{\int_{d_{min}}^{D_{max}} f(t)dt}{\int_{d_{min}}^{d_{max}} f(t)dt} \qquad (2)$$

최적의 2차 우선 순위 방식은 메인큐에서 거리가 0인 노드쌍 중 P(Dmax)값이 더 큰 노드 쌍에게 더 높은 우선 순위를 주는 방식이 된다.

그런데 P(Dmax)를 구하려면 2가지 문제를 해결해야 한다. 첫째는 알고리즘 수행 이전에 Dmax값을 알아야 한다는 점과, 둘째는 주어진 각 노드쌍에 대하여 거리분포함수 f(t)를 구하여야 한다는 점이다. 본 논문에서는 이 문제들을 다음과 같이 해결한다.

첫째, 알고리즘 수행 이전에 K에 대한 한계거리값 Dmax를 정확히 알기는 불가능하다. 따라서 정확한 Dmax 값 대신에 Dmax에 대한 예측값 eDmax를 사용할 수 있다. [1]에서는 데이타의 균일 분포 가정하에서 예측값을

$$eDmax = \sqrt{K \times \rho} \quad (\text{단}, \rho \simeq \frac{area(R \cap S)}{\pi \times |R| \times |S|}) \qquad (3)$$

와 같이 제안하였다. 본 논문에서는 이 기법을 그대로 사용하기로 한다.

둘째, 정확한 거리분포함수 f(t) 대신에 이를 추상화한 분포함수를 사용한다. 즉, (그림 5)의 (b)의 거리분포함수 형태에서 x축 상의 최소거리지점(d_{min})과 최대 거리지점(d_{max})에서 중심점 거리지점(d_{center})으로 연결선을 그으면 분포함수 f(t)는 간단한 직선 함수의 조합으로 간략화된다. 여기서 중심점 거리(d_{center})는 양쪽 노드의 중심점 간의 거리로 구할 수 있다. 이때, 확률 P(Dmax)는 다음 (표 1)로 풀이된다.

(표 1)에서 첫 번째와 마지막 조건일 때는 각각 거리값이 Dmax 안에 존재하는 자식 노드쌍이 전혀 없거나 모든 자식 노드쌍이 Dmax 안에 존재함을 나타낸다. 이때에는 추가적인 우선 순위로서 노드쌍의 최대거리값이

표 1 추상화된 P(Dmax)

P(Dmax)	조건
0	$D_{max} < d_{min}$
$\frac{(D_{max} - d_{min})^2}{(d_{center} - d_{min})(d_{max} - d_{min})}$	$d_{min} \le D_{max} < d_{center}$
$\frac{d_{center} - d_{min}}{d_{max} - d_{min}} + (\frac{D_{max} - d_{center}}{d_{max} - d_{min}})(1 + \frac{d_{max} - D_{max}}{d_{max} - d_{center}})$	$d_{center} \le D_{max} < d_{max}$
1	$D_{max} \ge d_{max}$

작은 순서로 우선 순위를 부여한다.

5. 성능 평가

실험에 사용한 컴퓨터는 Solaris 2.7이 탑재된 Sun UltraSparc-II, 메인메모리 256M, Ultra 10 EIDE 인터페이스로 연결된 9Gbytes 하드디스크의 규격이다. 운영체제의 캐시 효과를 없애기 위해서 Solaris의 Direct I/O 기능을 사용하였다. 실험에 사용한 데이타는 미국 Bureau of Census에서 제공하는 Tiger/Line 97 지리정보 데이타 중 미국 아리조나 주의 도로 및 기타 공간 지역 객체를 나타내는 각각 633,461개와 189,642개의 실제 직선 데이타들이며, 이를 R*-트리[6]로 구성하였다. 거리 조인 알고리즘으로는 [1,2]에서 제안한 B-KDJ을 사용하였으며, K는 1에서 100,000까지 변화시켰다.

실험 대상 기법은 2차 우선 순위가 없는 경우(None), 2절 관련 연구에서 언급한 깊이 우선(Depth), 넓이 우선(Area), 3절에서 소개한 겹침 비율 우선(Overlap), 최대거리 우선(MaxDist), 그리고 본 논문이 제안하는 최적화 기법(ApprOpt) 등이다.

(표 2)는 각 2차 우선 순위 기법을 B-KDJ에 적용했을 때의 메인큐에 노드쌍 삽입 횟수를 측정한 것이다. 표에 나와 있는 바와 같이 넓이 우선 방식이 다른 방식에 비해서 두드러지게 좋지 않은 성능을 나타냈다. 이하의 실험 결과에서는 Area 방식을 제외하였다.

표 2 각 2차 우선 순위 기법에 대한 B-KDJ의 큐 삽입 회수

방식 / K	None	Depth	Area	Overlap	MaxDist	ApprOpt
1	17,359	9,835	505,377	12,401	7,698	6,753
10	23,040	15,627	506,259	14,186	12,237	11,529
100	24,583	17,369	511,789	15,641	14,390	12,675
1,000	35,472	31,392	539,457	28,102	27,323	23,893
10,000	122,566	130,641	642,479	126,333	113,210	109,883
100,000	913,360	922,115	1,450,896	862,049	813,015	756,118

(그림 6)은 넓이 우선 방식을 제외한 나머지 방식들에 대하여 2차 우선 순위를 적용하지 않는 방식에 대하여 B-KDJ의 큐 삽입 회수를 1로 설정했을 때 각각의 방식에 대한 상대적인 큐삽입 회수 비율을 나타낸 것이다. 본 논문이 제안한 방식(ApprOpt)이 모든 K에 대해서 고루 가장 좋은 성능을 나타냄을 알 수 있다. 큐 삽입 회수 측면에서 최대 61%(k=1 일 때)에서 최소 15%(K=10,000 일 때) 사이의 성능 향상을 보였다. 또한 실험 결과로 보면, K값이 작을 수록 상대적으로 2차 우선 순위 기법의 효율이 좋다는 것을 확인할 수 있다. 특이한 점은 K=10,000 일 때, 깊이 우선 방식(Depth), 겹침 비율 우선 방식(Overlap) 등은 2차 우선 순위를 쓰지 않는 방식(None)보다도 더 좋지 않은 결과 값을 보여주는데, 이것은 해당 방식이 잘 적용되지 않는 노드쌍들이 많다는 사실을 보여주는 것이다. 이에 비하여 최대 거리 우선 방식(MaxDist) 혹은 본 논문이 제안하는 최적의 방식(ApprOpt)은 이런 경우에도 좋은 성능을 보여주고 있다.

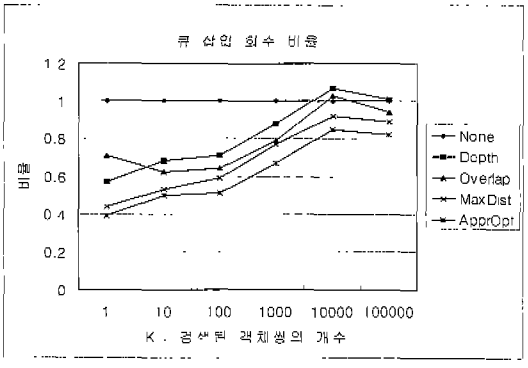


그림 6 각 2차 우선 순위의 B-KDJ에 대한 큐 삽입 회수 비율

(그림 7)은 각 2차 우선 순위에 대한 B-KDJ 알고리즘의 응답 시간을 비교한 것을 보여주고 있다. 마찬가지

로 2차 우선 순위를 쓰지 않는 때(None)의 수행 시간을 1로 하였을 때 각각의 기법에 대한 수행시간 비율을 표시하였다. 알고리즘의 응답 시간은 큐 삽입 회수 결과와 유사한 결과를 보여주었는데, 이는 큐 삽입 연산 비용이 거리조인 알고리즘의 성능에 매우 큰 영향을 미치는 요소로 작용하기 때문이다. 거리조인의 또 다른 비용 요소는 노드간의 거리 계산 회수, R-트리 노드 접근 등이 있는데 자세한 내용은 [1]에 기술되어 있다. 다만, 2차 우선 순위 기법은 주로 큐 삽입 연산 비용에 영향을 미친다. 실험 결과(그림 7)에서는 최대 58%(K=1 일 때)에서 최소 13%(K 10,000 일 때)까지 본 논문이 제안하는 기법이 성능을 향상시키고 있음을 보여주었다.

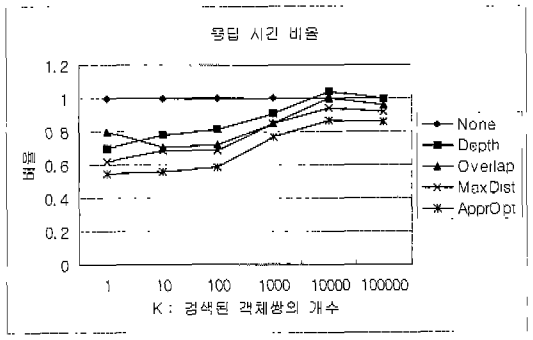


그림 7 각 2차 우선 순위의 B-KDJ에 대한 응답 시간 비율

6. 결 론

다차원 데이타에 대한 거리조인 처리 기법은 지리정보 시스템, 이미지 검색 시스템, 데이타 마이닝 등에서 기본 연산으로서 유용하게 사용될 수 있다.

본 논문에서는 최근에 제안된 공간 조인 기법인 거리조인에서 사용되는 우선 순위 큐의 동점자 노드쌍들에 대한 최적화된 우선 순위 처리 기법을 제안하였다. 동점자 노드쌍들은 매우 빈번하게 발생하며, 이들 간의 우선 순위는 거리조인 알고리즘의 성능에 많은 영향을 미칠 수 있다. 기존의 제한적인 기법들에 비하여, 특히 본 논문에서는 어떤 경우의 기하학적인 관계에 있는 노드쌍들에 대해서도 최적인 우선 순위를 부여할 수 있는 기법을 제안하고 있다. 제안된 기법을 통해서 거리조인 알고리즘의 전체 수행시간이 최고 58%까지 단축되었다.

거리조인은 R-트리를 기반으로 하고 있기 때문에, 대략 10차원 이상의 고차원 데이타에 대한 적용이 어렵다. 따라서 고차원에서의 거리조인에 대한 후속 연구가 필요하다.

참 고 문 헌

[1] Hyoscop Shin, Bongki Moon, and Sukho Lee, "Adaptive Multi-Stage Distance Join Processing," Proc. of 2000 ACM-SIGMOD Conference, pages 343-354, Dallas, TX, May 2000.

[2] 신효섭, 이석호, "공간 데이타베이스에서 최근접 K쌍을 찾는 효율적 기법," 정보 과학회 논문지: 데이타베이스, 27권 2호, pp.238-246, 2000. 6.

[3] Gisli R. Hjaltason, and Hanan Samet, "Incremental Distance Join Algorithms for Spatial Databases," Proc. of 1998 ACM-SIGMOD Conference, pages 237-248, Seattle, WA, June 1998.

[4] Antonio Corral, Yannis Manolopoulos, Yannis Theodoridis, and Michael Vassilakopoulos, "Closest Pair Queries in Spatial Databases," Proc. of 2000 ACM-SIGMOD Conference, pages 189-200, Dallas, TX, May 2000.

[5] Antonin Guttman, "R-Trees: A Dynamic Index Structure for Spatial Searching," Proc. of 1984 ACM-SIGMOD conference, pages 47-57, Boston, MA, June 1984.

[6] Norbert Beckmann, Hans-Peter Kriegel, Ralf Schneider, and Bernhard Seeger, "The R*-tree: An Efficient and Robust Access Method for Points and Rectangles," Proc. of 1990 ACM-SIGMOD conference, pages 322-331, Atlantic City, NJ, May 1990.

[7] Timos K. Sellis, Nick Roussopoulos, and Christos Faloutsos, "The R+-Tree: A Dynamic Index for Multi-Dimensional Objects," Proc. of 1987 VLDB conference, pages 507-518, Brighton, England, September 1987.

[8] Stefan Berchtold, Daniel A. Keim, and Hans-Peter Kriegel, "The X-Tree: An Index Structure for High-Dimensional Data," Proc. of 1996 VLDB conference, pages 28-39, Mumbai(Bombay), India, September 1996.

[9] Andreas Henrich, "The LSD^h-tree: An Access Structure for Feature Vectors," Proc. of 1998 ICDE conference, pages 362-369, Orlando, FL, February 1998.

[10] John T. Robinson, "The K-D-B-tree: A Search Structure for Large Multidimensional Dynamic Indexes," Proc. of 1981 ACM-SIGMOD conference, pages 10-18, Ann Arbor, Michigan, April 1981.

신 효 섭

1994년 서울대학교 컴퓨터공학과 학사. 1996년 서울대학교 컴퓨터공학과 석사. 2002년 서울대학교 전기.컴퓨터공학부 박사 졸업 예정. 1999년 ~ 2001년 2차례 미국 아리조나 주립대 전산학과 방문 연구. 2001년 7월 ~ 현재 삼성전자 CTO 전략실 소프트웨어 센타 책임 연구원. 관심분야는 공간 데이타베이스, 고차원 데이타베이스, 임베디드 데이타베이스, XML 등

문 봉 기

1983년 서울대학교 컴퓨터공학과 학사. 1985년 서울대학교 컴퓨터공학과 석사. 1985년 ~ 1990년 삼성전자 및 삼성종합기술원 근무. 1996년 University of Maryland College Park 전산학 박사. 1997년 ~ 현재 University of Arizona 전산학과 조교수 재직. 관심분야는 Spatial/Multidimensional databases, XML, Scalable web server, Moving objects, Bioinformatics, Parallel processing.

이 석 호

1964년 연세대학교 정치외교학과 졸업. 1975년, 1979년 미국 텍사스대학교 전산학 석사와 박사학위 취득. 1979년 ~ 1982년 한국과학원 전산학과 조교수. 1982년 ~ 1986년 한국정보과학회 논문편집위원장. 1986년 ~ 1988년 한국정보과학회 부회장. 1988년 ~ 1989년 미국 IBM T.J. Watson 연구소 객원교수. 1988년 ~ 1990년 데이타베이스연구회 운영위원장. 1989년 ~ 1991년 서울대학교 중앙교육연구전산원 원장. 1994년 한국정보과학회 회장. 1997년 ~ 현재 한국학술진흥재단 부설 첨단학술정보센터 소장. 1982년 ~ 현재 서울대학교 컴퓨터공학부 교수. 관심분야는 데이타베이스, 멀티미디어 데이타베이스