DOI: 10.3745/KIPSTD.2008.15-D.4.451

능동적 재조정: TPR*-트리의 검색 성능 개선 방안

김 상 욱[†] · 장 민 희[††] · 임 성 채[†††]

요 약

최근 들어, 이동 객체의 위치 정보를 이용한 응용의 등장으로 시공간 데이터베이스를 위한 인덱스 기법의 중요성이 점차 커지고 있다. TPR*-트리는 미래 시간 질의의 효율적인 처리를 위하여 가장 널리 사용되는 인덱스 구조이다. TPR*-트리는 CBR(conservative bounding rectangle)의 개념을 이용하여 이동 객체들의 미래 위치의 범위를 추정하는 방식을 사용한다. 그러나 CBR은 시간이 지남에 따라 지나치게 확대됨으로써 질의 처리 성능을 크게 저하시키는 문제를 야기시킨다. 본 논문에서는 능동적인 CBR 재조정을 통하여 이러한 CBR의 지나친 확대를 방지하고, 이 결과 TPR*-트리의 질의 처리 성능을 개선할 수 있는 효과적인 기법을 제안한다. 제안한 기법은 질의 처리를 위하여 TPR*-트리의 단말 노드를 액세스한 시점에 CBR 재조정의 필요여부를 점검하도록 함으로써 이러한 점검을 위한 추가적인 디스크 액세스 비용을 요구하지 않는다. 또한, CBR의 재조정이 필요한가의 여부를 판정하기 위하여 재조정을 위한 추가 비용과 향후의 질의 비용을 모두 고려하는 새로운 비용 모델을 정립한다. 제안된 기법을 통하여 갱신이 자주 발생하지 않는 경우에도 CBR의 비정상적인 확대를 방지할 수 있다. 제안된 기법의 성능 개선 효과를 정량적으로 검증하기 위하여 다양한 실험을 수행한다. 실험 결과에 의하면, 제안된 기법은 질의 처리 시 기존 기법과 비교하여 최대 40%이상의 성능 개선 효과를 보인다.

키워드 : 이동 객체, 시공간 데이터베이스, 미래 예측, 인덱스

Active Adjustment: An Approach for Improving the Search Performance of the TPR*-tree

Sang-Wook Kim[†] · Min-Hee Jang[††] · Sungchae Lim[†††]

ABSTRACT

Recently, with the advent of applications using locations of moving objects, it becomes crucial to develop efficient index schemes for spatio-temporal databases. The TPR*-tree is most popularly accepted as an index structure for processing future-time queries. In the TPR*-tree, the future locations of moving objects are predicted based on the CBR(Conservative Bounding Rectangle). Since the areas predicted from CBRs tend to grow rapidly over time, CBRs thus enlarged lead to serious performance degradation in query processing. Against the problem, we propose a new method to adjust CBRs to be tight, thereby improving the performance of query processing. Our method examines whether the adjustment of a CBR is necessary when accessing a leaf node for processing a user query. Thus, it does not incur extra disk I/Os in this examination. Also, in order to make a correct decision, we devise a cost model that considers both the I/O overhead for the CBR adjustment and the performance gain in the future-time owing to the CBR adjustment. With the cost model, we can prevent unusual expansions of BRs even when updates on nodes are infrequent and also avoid unnecessary execution of the CBR adjustment. For performance evaluation, we conducted a variety of experiments. The results show that our method improves the performance of the original TPR*-tree significantly.

Key Words : Moving Objects, Spatio-Temporal Databases, Future Prediction, Index

1. 서 론

최근 들어, GPS(global positioning system) 기술 및 이동 통신 기술의 발달로 인하여 이동 객체의 위치 정보를 효과적으로 활용하기 위한 방안에 대한 관심이 증대되고 있다. 이동 객체는 주기적으로 자신의 위치를 서버로 전송하는데, 이렇게 전송된 정보는 시간의 흐름에 따라 공간적인 위치가 변화하는 시공간 데이터(spatio-temporal data)의 특성을 갖는다[Lee02]. 이동 객체 데이터베이스(moving object database)는 이와 같이 시간의 흐름에 따르는 객체의 위치 변화 정보가 저장된 시공간 데이터베이스를 의미한다[Wol98].

이동 객체 데이터베이스에 대한 사용자의 질의는 이동 객체의 과거의 움직임을 검색하는 과거 시간 질의(past-time

† 종신회원 : 한양대학교 정보통신대학 정보통신학부 교수
†† 준 회 원 : 한양대학교 정보통신학과 박사과정
††† 정 회 원 : 동덕여자대학교 컴퓨터전공 조교수
논문접수 : 2008년 3월 4일
수 정 일 : 1차 2008년 4월 24일
심사완료 : 2008년 4월 26일

query)[Xu90][The96]와 이동 객체의 미래의 움직임을 예측하여 검색하는 미래 시간 질의(future-time query)의 두 가지 형태로 크게 분류할 수 있다[Sis97]. 이 중에서, 미래 시간 질의는 위치기반 서비스, 교통 정보 시스템, 항공기 통제 시스템 등 미래 상황 예측에 기반한 다양한 서비스에 활용이 가능하다[Lee02]. 대표적인 미래 시간 질의의 예로는 "1:00 pm에 한남 대교를 지날 것으로 보이는 모든 차량들을 검색하라"를 들 수 있다.

현재까지, 인덱스를 이용하여 미래 시간 질의를 보다 빠르고 효과적으로 처리하기 위한 다양한 방법들이 연구되어 왔다[Mok03]. 미래 시간 질의를 위한 대표적인 인덱스로는 VCI-트리[Pra02], TPR-트리[Sal00], TPR*-트리[Tao03] 등이 있으며, 이들 중 TPR-트리의 단점을 개선한 TPR*-트리가 가장 좋은 질의 처리 성능을 보이는 것으로 알려져 있다.

TPR*-트리는 R*-트리[Bec90]를 기본 구조로 가지는 인덱스 기법으로, CBR(conservative bounding rectangle)이라는 개념을 사용하여 이동 객체들을 관리한다. CBR은 각 노드의 이동 객체들이 위치하는 공간인 MBR(minimum bounding rectangle)[Bec90]과 그 노드의 이동 객체들이 공간상의 각 축에서 갖는 속도 중 최대, 최소값을 저장하여 이를 이동 객체의 미래 위치를 검색하기 위한 정보로 이용한다. 이러한 특성 때문에 인덱스의 생성 이후에 시간이 흐름에 따라 사장 영역(dead space)[Bec90]이 증가한다는 문제점을 가지고 있다. 사장 영역이란 노드의 영역 내부에서 실제 이동 객체가 존재하지 않는 공간을 의미하는데 이런 사장 영역이 증가함에 따라 노드에 대응되는 영역들 간에 중복이 심화된다. TPR*-트리 내의 사장 영역과 중복 영역의 증가는 인덱스 검색 시에 액세스되는 노드들의 수를 증가시키고 이 결과 질의 처리의 급격한 성능 저하를 초래한다[Tao03].

TPR*-트리에서는 질의 처리 성능의 저하를 방지하기 위하여 이동 객체의 위치 정보 갱신 시에 맞춰 이동 객체들의 실제 위치를 반영하여 CBR을 재조정하는 기법을 사용한다[Tao03]. 그러나 이동 객체의 위치 정보 갱신이 일어나기 전까지는 CBR내에 포함되어 있는 최대, 최소 속도값으로 인하여 사장 영역이 지속적으로 증가한다. 따라서 기존 기법으로는 질의 처리 성능 저하를 방지하는 데 한계가 있다.

본 논문에서는 사장 영역을 감소시켜 질의 처리 성능을 크게 완화시킬 수 있는 효과적인 CBR 재조정 기법에 대하여 논의한다. TPR*-트리의 기존 CBR 재조정 기법은 갱신이라는 제한된 연산 시에만 이루어지기 때문에 사장 영역의 지속적인 증가 문제가 해결되지 않는다. 본 논문에서는 질의 처리 시 액세스 되는 노드가 디스크에서 메모리 버퍼에 적재된다는 점을 이용하여 질의 처리 시 액세스된 노드의 CBR을 재조정 해주는 새로운 기법을 제안한다. 본 논문에서는 이러한 기법을 능동적 재조정(active adjustment) 기법이라 부른다. 능동적 재조정 기법은 메모리 버퍼 상에 존재하는 노드 정보를 기반으로 CBR 재조정 필요 여부를 판단하므로 추가적인 디스크 액세스 비용을 요구하지 않는다. 또한, 기존의 기법과는 달리 매우 빈번하게 발생하는 사용자 질의 연산 시에도 CBR 재조정을 수행할 수 있다. 따라서 비정상적인 CBR의 증가 여부를 지속적으로 감시하므로 전체적인 사장 영역을 크게 감소시킬 수 있다. 저자들이 아는 한 이와 같이 질의 처리 시 액세스된 노드의 사장 영역을 감소시켜 주는 능동적 재조정 기법은 미래 시간 질의를 위한 연구 중 최초의 시도이다.

전술한 바와 같이, 능동적 재조정 기법은 CBR 재조정 필요 여부의 판단을 위하여 추가적인 디스크 액세스 비용을 요구하지 않는다. 그러나 이러한 판단에 따라 노드의 CBR을 실제로 재조정 했을 시에는 디스크에 이 정보를 반영해야 한다. 따라서 불필요한 CBR 재조정은 추가적인 디스크 액세스를 유발시키게 되므로 노드의 CBR 재조정 여부를 올바르게 판단할 수 있는 기준이 필요하다.

본 논문에서는 CBR 재조정을 위해서 필요한 추가 비용과 재조정을 통해 이후의 질의 처리에서 얻을 수 있는 비용의 절감을 모두 고려하여 CBR 재조정 여부를 판단하는 비용 모델을 제시한다. 이러한 비용 모델을 통하여 전체적인 성능 향상의 정도를 미리 계산할 수 있기 때문에 불필요한 CBR 재조정을 사전에 방지할 수 있다.

본 논문에서는 다양한 실험을 이용한 성능 분석을 통하여 제안하는 기법의 우수성을 규명한다. 실험 결과에 의하면, 능동적 재조정 기법은 기존 TPR*-트리의 성능을 최대 40% 이상 개선시키는 효과를 보인다.

본 논문의 구성은 다음과 같다. 제 2장에서는 관련 연구로서 기존 TPR-트리와 TPR*-트리의 특성 및 장단점에 관하여 논의하고, CBR 재조정의 개념에 대해 설명한다. 제 3장에서는 본 연구에서 제안하는 능동적 재조정 기법의 기본 개념, 비용 모델, 그리고 제안하는 알고리즘에 관하여 상세히 서술한다. 제4장에서는 성능 평가를 통하여 능동적 재조정 기법의 우수성을 규명한다. 끝으로, 제 5장에서는 논문을 요약하고 결론을 맺는다.

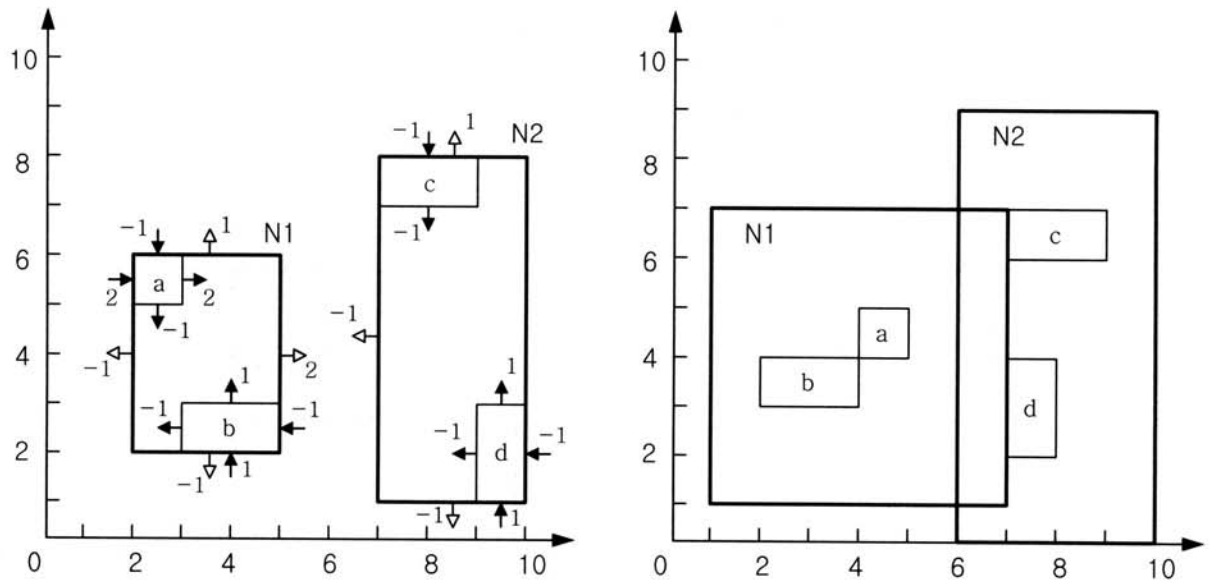
2. 관련 연구

본 장에서는 관련 연구로서 미래 예측을 위한 대표적 인덱스 구조인 TPR-트리와 TPR*-트리에 대해 기술하고, 그 장단점에 관하여 논의한다.

2.1 TPR-트리

TPR-트리[Sal00]는 R*-트리[Bec90]의 구조를 기반으로 고안된 것으로서, 이동 객체의 특정 시각에서의 위치와 속도 정보를 저장함으로써 이동 객체의 미래 위치를 검색할 수 있는 인덱스 구조이다. 이동 객체들의 미래 위치 예측을 위해서 R*-트리에서 사용하는 MBR(minimum bounding rectangle) 대신 CBR(conservative bounding rectangle)을 사용하여 이동 객체의 위치를 인덱싱한다.

CBR은 현재 특정 시각에서의 이동 객체들의 영역을 표현하는 MBR과 이 MBR 내부의 이동 객체들의 각축에 대한 최대, 최소의 속도로 구성된다. 미래의 임의의 시점에 이동 객체가 위치할 수 있는 인덱스 공간의 범위는 CBR에 저장



(그림 1) TPR-트리에서 미래 예측을 위한 CBR의 확장

된 MBR의 위치 정보와 최대, 최소 속도 정보를 이용한 계산에 의해 예측된다. 또한, 각 CBR의 MBR과 이동 객체들의 최대, 최소 속도를 통해 계산되어 예측된 인덱스 공간 영역을 해당 노드의 BR(bounding rectangle)이라 정의한다. CBR이 내부의 모든 객체들의 이동을 대표하는 값으로서 각 축의 최대, 최소의 속도 값만을 저장하는 이유는 그 CBR의 하위 노드들에 저장된 모든 이동 객체들에 대하여 위치와 속도 값을 유지함으로서 발생하는 지나친 저장 공간의 오버헤드를 피하기 위해서이다.

(그림 1)은 TPR-트리에서 미래 예측을 위한 CBR의 확장을 나타낸다. (그림 1(a))에서 a, b, c, d는 각각 이동 객체를 의미하고, N1, N2는 이동 객체들을 저장하는 노드와 대응되는 MBR을 의미한다. 검은 화살표와 숫자는 이동 객체의 각축에 대한 최대, 최소 이동 속도를 나타낸다. 하얀 화살표는 해당 MBR 내의 객체들의 각축에 대한 최대, 최소 속도 중 최대, 최소를 나타낸다. CBR은 이 MBR과 하얀 화살표로 구성된다. (그림 1(a))는 인덱스가 구성된 time 0 시점에서 노드에 저장되어 있는 CBR을 나타낸다. (그림 1(b))는 각 이동 객체와 CBR의 이동을 고려하여 확장된 time 1 시점의 BR을 나타낸다.

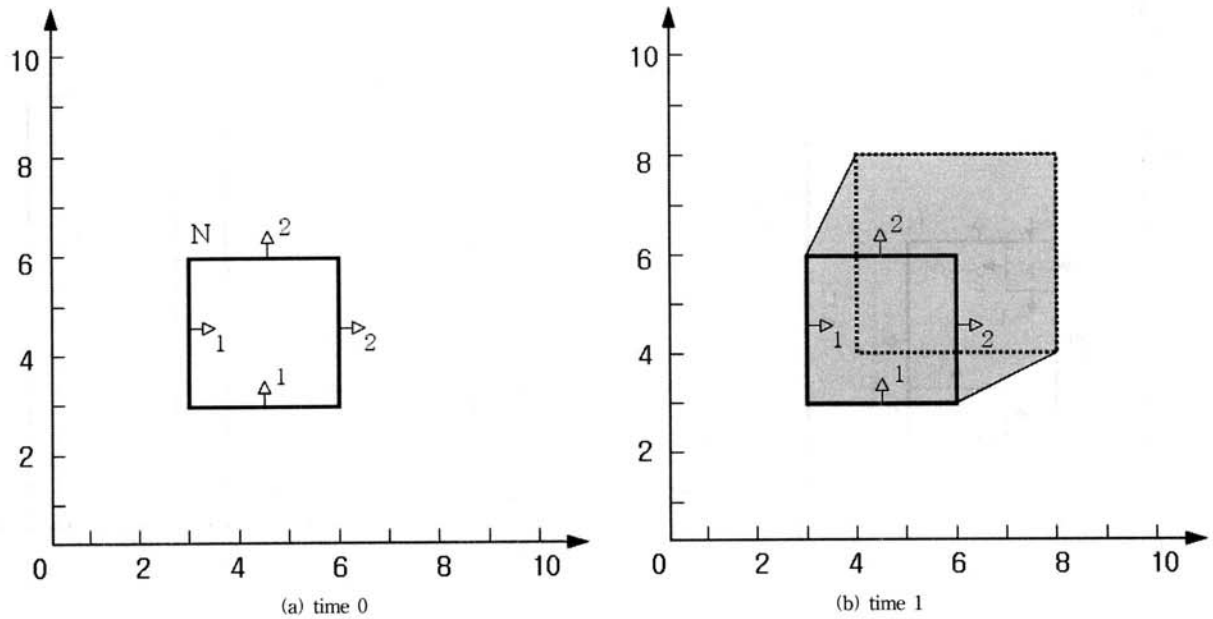
TPR-트리는 사용자의 질의가 들어오면 CBR에 저장되어 있는 정보를 바탕으로 질의 시간에 맞춰 BR을 확장한 후, 확장된 BR을 토대로 탐색할 하위 노드를 결정한다. 이 때, 이동 객체들 중 최대, 최소 속도만을 가진다는 CBR의 특성에 의해서 (그림 1)에서 보는 바와 같이 BR은 시간의 흐름에 따라 넓이가 지속적으로 커지는 경향을 보인다. 이처럼 CBR을 이용한 인덱싱은 BR의 크기가 항상 커진다는 단점으로 인하여 사장 영역이 크게 증가한다. 따라서 각 노드에 대응하는 영역간의 중복이 심화되고 이로 인하여 질의 처리 시에 읽어 들여야 할 노드의 개수가 크게 증가하는 문제점이 있다. TPR-트리에서는 BR이 무한대로 확장하는 것을 방지하기 위해서 이동 객체의 위치 정보 갱신 시에 객체의 실제 위치를 반영하여 CBR을 재조정하는 정책을 사용한다.

2.2 TPR*-트리

TPR*-트리[Tao03]는 기본적으로 TPR-트리와 동일한 구조를 사용하며 질의 처리 방식도 동일하다. 그러나 갱신 연산 시에 TPR-트리가 R*-트리의 알고리즘을 그대로 사용하는데 비하여 TPR*-트리에서는 객체의 이동성을 고려한 향상된 알고리즘을 사용한다. 이로 인해 TPR*-트리는 TPR-트리에 비하여 갱신 및 검색 성능을 개선할 수 있다.

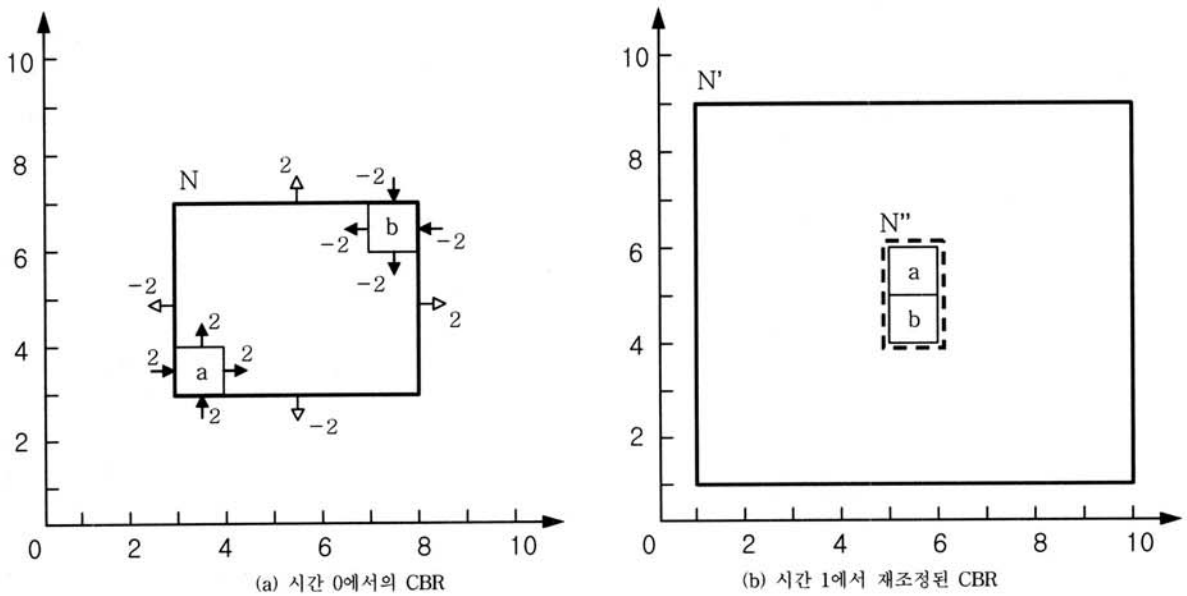
삽입 알고리즘을 예로 들면, TPR-트리는 이동 객체 삽입 시점에서만의 MBR의 면적, 둘레, 겹침, 거리를 고려하여 이동 객체가 삽입될 노드들을 결정한다. 이런 방식은 고정된 객체의 위치 인덱싱에는 매우 효과적일 수 있지만, 이동이란 특성을 가지고 있는 이동 객체에 대해서는 문제점을 안고 있다. 따라서 단순히 특정 시점에서의 노드의 영역을 볼 것이 아니라 시간에 따른 노드의 움직임인 BR을 고려한 삽입 알고리즘이 필요하다.

이런 판단에 의해 TPR*-트리에서는 특정한 미래 시간까지 BR이 확장해 가는 공간인 스위핑 영역(sweeping region)의 크기 변화를 기준으로 삽입 연산을 수행한다. 스위핑 영역이란, 서로 다른 두 시점 $t_1 < t_2$이 있을 때, t_1의 BR이 t_2의 BR로 이동 및 확장해 갈 때 쓸고 지나게 되는 공간을 의미한다. (그림 2)는 TPR*-트리의 스위핑 영역을 나타내는 예이다. (그림 2(a))는 시간 0에서 노드 N에 대한 BR을 표시하며, 시간의 흐름에 대한 BR의 확장 및 이동으로 (그림 2(b))와 같이 시간 1에서는 점선 사각형과 같이 확장 이동한



(a) time 0 (b) time 1

(그림 2) 스위핑 영역



(a) 시간 0에서의 CBR (b) 시간 1에서 재조정된 CBR

(그림 3) CBR 재조정에 따른 BR 크기의 축소

다. (그림 2(b))에서 검게 칠해진 영역은 시간 0에서 시간 1까지 노드 N의 BR이 확장하며 쓸고 지난 스위핑 영역을 표시하고 있으며, 이 스위핑 영역의 넓이는 23이다.

TPR*-트리에서는 삽입 연산 시 루트 노드에서부터 하위 노드로 내려가며 스위핑 영역의 크기를 계산하여 그 값을 최소로 하는 방향으로 노드를 선택하여 이동 객체를 삽입한다. 이러한 전략으로 인하여 TPR*-트리는 TPR-트리의 갱신 알고리즘보다 많은 갱신 비용을 요구하기도 한다. 그러나 인덱스 구조를 TPR-트리에 비해 보다 잘 구성하기 때문에 전체적인 질의 처리 성능을 크게 개선할 수 있다.

이 이외에도 TPR-트리보다 향상된 삭제 알고리즘을 사용하여 삽입, 삭제를 포함한 갱신 연산에서 월등한 성능 향상 효과를 보인다. TPR*-트리의 실험 결과에 의하면 TPR*-트리는 TPR-트리에 비해 최대 5배 정도의 성능 개선을 보인다[Tao03].

3. 능동적 재조정 기법

본 장에서는 TPR*-트리의 검색 성능을 효과적으로 개선할 수 있는 방법에 대해서 제안한다. 먼저, 본 연구를 수행하게 된 동기에 관하여 기술하고, 제안하는 기본 전략과 알

고리즘에 관하여 살펴본다.

3.1 연구 동기

TPR*-트리의 CBR은 내부의 객체들이 공간상의 각 축에서 갖는 속도들 중 최대, 최소값만을 저장하기 때문에 각 노드의 BR은 지속적으로 확장한다. 이로 인하여, 시간이 흐를수록 각 BR의 면적이 증가하게 되므로 사장 영역이 증가하고 이에 따라 노드간의 영역 중복 또한 심화된다. 이 결과, 인덱스 검색을 위한 디스크 액세스 수가 증가 되어 질의 처리 성능이 떨어지게 된다. TPR*-트리는 이러한 문제를 해결하기 위하여 갱신 시에 이동 객체의 실제 위치를 반영하여 CBR을 재조정하는 정책을 사용한다.

(그림 3)은 CBR 재조정의 효과를 나타내는 것이다. (그림 3(a))의 이동 객체 a, b는 각각의 속도 값에 의해서 time 1 시점에서 (그림 3(b))의 N''으로 위치가 변화한다. TPR*-트리의 CBR은 객체들이 공간상의 각 축에서 갖는 최대, 최소 속도 값만을 가지므로 MBR N은 time 1시점에서 N'과 같이 크게 확장된다.

(그림 3(b))에서 time 1 시점에서 BR N'은 많은 사장 영역을 포함하게 된다. 만일, time 1 시점에서 이동 객체의 위치 정보가 갱신된다면, TPR*-트리의 CBR재조정 기법에 의해서 BR의 크기를 N'에서 N''으로 줄일 수 있다. 이 결과, 사장 영역과 중복 영역이 감소하게 되므로 전체 질의 처리 성능을 개선시킬 수 있다.

그러나 TPR*-트리의 기존 CBR 재조정은 갱신이라는 제한된 연산의 수행 시에만 발생하게 된다. 따라서 특정 노드에서 갱신이 일어난 후 다음 갱신이 일어나기 까지는 CBR의 재조정이 발생하지 않는다. 또한, 갱신이 특정 지역에 편중되는 경우에는 갱신이 일어나지 않는 지역에서 재조정이 거의 일어나지 않는다. 따라서 기존 CBR 재조정 기법은 시간이 갈수록 사장 영역과 중복 영역이 증가하게 되며, 이로 인하여 전체적인 질의 처리 성능이 떨어지게 된다. 본 연구의 목적은 TPR*-트리의 기존 CBR 재조정 기법의 이러한 문제점을 해결하여 전체 질의 처리 성능의 저하을 개선시키는 새로운 기법을 제안하는 것이다.

3.2 제안하는 기본 전략

본 절에서는 TPR*-트리의 문제점인 사장 영역을 감소시켜 질의 처리 성능을 크게 개선시킬 수 있는 효과적인 CBR 재조정 기법에 대하여 자세히 서술한다.

제안하는 기법의 설명을 위해 임의의 질의가 TPR*-트리의 탐색을 통해 단말 노드 N에 접근하는 상황을 가정하자. 단말 노드 N에 j 번째 갱신 연산이 수행된 시점을 Tu_j, 그리고 $j+1$ 번째 갱신 연산이 수행된 시점을 Tu_{j+1}이라고 하자. 이 두 시점들 사이에서 단말 노드 N에 탐색해 내려가는 i번째 질의를 $Q_{j,i}$로, 질의 $Q_{j,i}$의 처리를 위해 N에 접근한 시점을 $Tq_{j,i}$ 로 하자. 이 갱신 구간 $[Tu_i, Tu_{j+1}]$ 사이에 k 개의 질의가 처리되었다면 다음과 같은 시퀀스가 성립한다.

$$Tu_j < Tq_{j,1} < Tq_{j,2} < Tq_{j,3} < \cdots < Tq_{j,k} < Tu_{j+1}$$

기존의 방식에서는 이동 객체 갱신 연산 시에만 CBR 재조정을 수행하기 때문에 갱신 구간 $[Tu_i, Tu_{j+1}]$ 사이에서는 N의 탐색 경로에 관한 CBR 재조정이 일어나지 않아 사장 영역이 증가한다. 만약, 이 갱신 구간 내에서 CBR 재조정을 수행할 수 있다면 사장 영역을 크게 감소시킬 수 있어 전체 질의 처리 성능을 개선시킬 수 있다. 본 논문에서는 질의가 들어오는 시점인 $Tq_{j,i}$에 루트에서 단말 노드 N까지의 탐색 경로내의 노드들이 메모리 버퍼에 올라온다는 점에 착안하여 능동적으로 CBR 재조정을 수행함으로써 질의 처리 성능을 개선시키는 방법을 제안한다. 본 논문에서는 이러한 기법을 능동적 재조정 기법(active adjustment) 이라 부른다.

능동적 재조정 기법은 메모리 버퍼 상에 존재하는 노드 정보를 기반으로 CBR 재조정 필요 여부에 대한 판단을 수행하기 때문에 추가적인 디스크 액세스 비용을 요구하지 않는다. 그러나 이러한 판단에 따라 노드의 CBR 을 실제로 재조정 했을 시에는 디스크에 이 정보를 반영하기 위한 추가적인 디스크 액세스 비용이 요구된다. 따라서 불필요한 CBR 재조정은 추가적인 디스크 액세스를 유발시키게 되므로 노드의 CBR 재조정 여부를 올바르게 판단할 수 있는 기준이 필요하다.

본 논문에서는 CBR 재조정을 위해서 필요한 추가 비용과 재조정을 통해 이후의 질의 처리에서 얻을 수 있는 비용의 절감을 모두 고려하여 CBR 재조정 여부를 판단하는 비용 모델을 제시한다. 이러한 비용 모델을 통하여 전체적인 성능 향상의 정도를 미리 계산할 수 있기 때문에 불필요한 CBR 재조정을 방지할 수 있다.

비용 모델을 정립하기 위해서 중요한 것은 스위핑 영역의 크기이다. 하나의 CBR에 대한 스위핑 영역의 크기는 전체 공간상에서 임의의 한 질의가 들어왔을 때 그 CBR과 대응되는 노드를 액세스할 확률을 의미한다[Tao03]. 이 스위핑 영역은 노드와 대응되는 CBR의 면적과 속도를 기반으로 계산된다. 따라서 CBR의 재조정은 스위핑 영역의 크기를 줄여주는 효과를 가져 오게 되므로 질의와 상관없는 노드를 액세스 할 확률을 낮춰 전체 질의 처리 성능을 개선시키게 된다. $Tq_{j,i}$이전의 CBR을 그대로 유지하였을 때 Tu_{j+1}시점에서의 스위핑 영역의 크기를 SR이라고 하고, $Tq_{j,i}$에서 재조정된 CBR을 이용하여 계산된 Tu_{j+1}시점에서의 스위핑 영역의 크기를 SR'이라고 하자. 이때, SR과 SR'의 차이(=SR-SR')를 CAB라 정의한다. CAB의 크기가 클수록 능동적 CBR 재조정을 통한 질의 처리 성능이 크게 개선됨을 의미한다. 즉, CAB란 능동적 CBR 재조정을 통하여 해당 노드가 질의에 의하여 디스크로부터 액세스될 확률이 얼마나 줄어드는가를 정량적으로 측정함을 의미한다. 계산한 CAB 값이 CBR 재조정에 따른 갱신 비용에 비해 큰 값을 가진다면 $Tq_{j,i}$시점에 능동적 CBR 재조정을 수행한다.

CAB 계산에 고려되어야 할 요소는 단위 시간 당 질의 빈도이다. 앞에서 언급한 CAB의 계산은 하나의 질의가 들어왔을 때 질의 처리의 성능 상 이익을 측정한 것이다. 따라서

비용 모델을 올바르게 정립하기 위해서는 하나의 질의가 아닌 $[Tu_i, Tu_{j+1}]$ 사이에 들어오는 질의들에 대하여 CAB를 계산해야 한다. Tu_{j+1} 시간은 이동 객체의 갱신 연산에 의해 CBR 재조정이 일어나는 시간이기 때문에 이전까지의 단위 시간 당 질의 빈도를 고려하여 CAB를 계산한다.

Tu_{j+1} 시간까지의 단위 시간 당 질의 빈도를 고려하기 위해서는 갱신 연산 시점인 Tu_{j+1}을 알아야 한다. 이 시점은 향후 갱신 연산이 발생하는 시점으로 $Tq_{j,i}$ 시점에서는 이를 정확히 알 수 없다. 따라서 제안하는 기법에서는 이동 객체 당 평균 갱신 주기와 단말 노드 당 평균 이동 객체 개수를 이용하여 각 단말 노드에 발생하는 갱신 연산의 평균 주기 P_u를 계산한다. 이 갱신 주기를 이용하여 $Tq_{j,i}$ 시점에서 $Tu_j + P_u$를 다음 노드 갱신의 발생 시점 Tu_{j+1}로 예측할 수 있다. 본 연구에서는 이 시점 $Tu_j + P_u$를 TS라는 하나의 타임스탬프 필드로 각 단말 노드에 저장함으로써 CAB 계산에 이용한다. 이와 같은 정의들에 따라 CAB의 계산은 아래와 같은 식으로 이루어진다.

$$CAB(Tq_{j,i}, Tu_j+Pu) = \left(\frac{SR(Tq_{j,i}, Tu_j+Pu) - SR'(Tq_{j,i}, Tu_j+Pu)}{2}\right) \times (Q_{freq} \times (Tq_{j,i}, Tu_j+Pu))$$

위 공식에서 CAB($Tq_{j,i}$, Tu_j+P_u)는 질의가 들어오는 시점에서 다음 갱신 시간까지 스위핑 영역의 변화로 얻게 되는 CAB이며, $SR(Tq_{j,i}, Tu_j+Pu)$는 질의가 들어온 시점에서 다음 갱신 시간까지 재조정되지 않은 CBR이 갖는 스위핑 영역의 크기, $SR'(Tq_{j,i}, Tu_j+Pu)$는 질의가 들어온 시점에서 재조정된 CBR이 다음 갱신 시간까지 갖는 스위핑 영역의 크기이다. 또, Q_{freq}는 단위 시간 1당 발생하는 질의의 빈도이고, 여기에 ($Tq_{j,i}$, Tu_j+Pu)를 곱하여 질의가 들어온 시점에서 다음 갱신 시간까지의 질의 빈도를 구한다. 위 공식에서 스위핑 영역 계산 시 $(SR(Tq_{j,i}, Tu_j+Pu) - SR'(Tq_{j,i}, Tu_j+Pu))$를 2로 나누어준 이유는 질의들에 해당하는 스위핑 영역의 평균적인 크기를 얻기 위해서이다. SR과 SR′은 모두 Tu_j+P_u 시점에서 스위핑 영역의 면적을 의미하는데 질의는 $[Tq_{j,i}, Tu_j+P_u]$시간에 연속적으로 들어온다. 따라서 Tu_j+P_u시점에서의 스위핑 영역에 질의 빈도를 곱해주는 것은 적당하지 않다. 정확한 계산을 위해서는 이후에 들어올 각각의 질의들에 맞는 스위핑 영역의 크기를 계산해야 하지만 들어오지도 않은 질의들에 대하여 스위핑 영역의 크기를 계산하는 것은 불가능하다. 따라서 본 연구에서는 질의들이 비교적 균등하게 들어온다고 가정하여, $[Tq_{j,i}, Tu_j+P_u]$시간에서 질의들에 해당하는 스위핑 영역의 평균적인 크기를 얻기 위해 Tu_j+P_u를 2로 나누었다.

능동적 재조정 기법의 판단에 따라 노드의 CBR 을 실제로 재조정하기 위해서는 디스크에 이 정보를 반영하기 위한 추가적인 디스크 액세스가 발생하게 된다. 따라서 불필요한 CBR 재조정은 오버헤드를 유발시키게 되므로 CBR 재조정 이익을 올바르게 판단할 수 있는 기준이 필요하다. 본 논문에서는 능동적 재조정에서 얻을 수 있는 비용의 절감 효과와 CBR 재조정으로 인한 오버헤드를 비교하여 능동적 재조정 기법을 수행하는 조건을 제안한다.

$$CAB(Tq_{j,i}, Tu_j+P_u) > H - 1 \quad \text{(조건 i)}$$

위의 (조건 i)는 능동적 재조정 기법을 수행하는 조건으로 CAB의 값이 H-1의 값을 넘을 때에만 CBR 재조정을 수행한다. H는 TPR*-트리의 높이이며, 능동적 CBR 재조정에 따른 갱신에 필요한 최대 디스크 액세스 횟수가 된다. 단, 능동적 CBR 재조정 시에 루트 노드는 항상 버퍼에 있기 때문에 트리 높이에서 1을 감소시켰다. TPR*-트리의 높이를 갱신 비용의 기준으로 둔 이유는 단말 노드의 CBR 재조정이 루트 노드의 CBR 정보까지 재조정되어야 하는 최악의 경우를 고려한 것이다.

한편, CBR의 능동적 재조정 여부의 판단 시에 디스크 액세스 비용은 발생하지 않지만, CPU 비용이 어느 정도 발생하게 된다. 따라서 매 질의마다 CBR 재조정 판단을 내리게 된다면 전체 질의 성능에 부정적인 영향을 미칠 수 있다. 따라서 다음과 같은 조건을 두어 불필요한 능동적 재조정 기법의 수행을 방지한다.

$$Tq_{j,i} + \text{epsilon} < Tu_j + P_u \quad \text{(조건 ii)}$$

위의 (조건 ii)는 질의가 들어온 시점이 다음 갱신 시간에 근접한 경우에는 CAB 계산 자체를 하지 않는다는 것이다. 질의가 들어온 시점 $Tq_{j,i}$가 Tu_j+P_u에 매우 가깝다면 CAB 계산을 수행하지 않고도 성능 향상 효과가 없거나 미미하다는 것을 알 수 있다. 따라서 불필요한 CAB 계산을 하여 전체 질의 성능을 저하시키는 상황을 피할 수 있다.

제안된 능동적 재조정 기법은 크게 세 가지 장점을 가진다. 첫째로, 질의 처리 연산은 갱신 연산에 비하여 발생 빈도가 크기 때문에 갱신 연산을 이용할 때보다 CBR 재조정을 자주 수행할 수 있다. 둘째, 갱신 연산과 더불어 질의 처리 연산 시에 CBR 재조정을 수행할 수 있으므로 갱신이 일어나지 않는 영역에서 재조정이 거의 일어나지 않는 현상을 막을 수 있다. 세째, 질의 처리 연산과 CBR 재조정 연산을 함께 처리하므로 CBR 재조정에 요구되는 디스크 액세스 비용이 적다.

3.3 알고리즘

본 절에서는 능동적 CBR 재조정을 위한 구체적인 알고리즘을 기술한다. 알고리즘은 유사 코드로 표현되며, 앞에서 사용한 기호들을 사용하여 기술한다. 제안하는 알고리즘은 기존 TPR*-트리의 삭제 및 삽입과 비교하여, 단말 노드 갱신 시에 해당 단말 노드의 TS 필드 값을 현재 시간으로 수정하는 것 외에는 차이가 없다. 따라서 본 논문에서는 삽입/삭제 알고리즘을 별도로 기술하지 않는다.

미래 시간 위치 질의는 (Q_t , Q_r)의 형태로 표현한다. Q_t는 질의의 미래 예측 시점을 표현하며, Q_r은 2차원 공간상에서 질의 영역의 범위를 표시한다. 즉, 주어진 시점 Q_t에 2차원 공간상의 영역 Q_r을 지날 것으로 예상되는 이동 객체를 찾고자 하는 것이다. 질의 조건에 부합되는 이동 객체들

```
Algorithm Search (Qr, Qt, rootNode, Sr) Input:
- (Qr, Qt): 미래 시간 위치 질의.
- rootNode: 탐색의 대상이 되는 서브트리의 루트 노드

Output:
- Sr: (Qr, Qt)의 결과에 해당하는 이동 객체들의 ID 집합.
 1: IF rootNode가 내부 노드일 경우 THEN
 2:    FOR rootNode의 각 엔트리 E에 대하여
 3:       IF E의 BR이 (Qr, Qt)와 교차하는 경우 THEN
 4:          CALL Search (Qr, Qt, childNode, childSr)
 5:          /* childNode: E가 가리키는 자식 노드 */
 6:          /* childSr: 재귀 함수 Search의 output */
 7:           IF childNode에 능동적 재조정에 의한 갱신이 일어났으면
THEN
 8:              E의 CBR을 재조정한다.
 9:          END IF
10:          Sr = Sr ∪ childSr
11:       END IF
12:    END FOR
13:    RETURN Sr
14: ELSE
15:    FOR rootNode의 각 이동 객체 O에 대하여
16:       IF Qt시점에 O의 위치가 (Qr, Qt)에 포함되는 경우 THEN
17:          Sr = Sr ∪ O의 id
18:       END IF
19:    END FOR
20:    IF Tqj,i + epsilon ≥ TS인 경우 THEN
21:       RETURN Sr
22:    END IF
23:    IF CAB(Tqj,i, Tuj+Pu) ≤ H - 1인 경우 THEN
24:       RETURN Sr
25:    ELSE 리프 노드 내의 객체들의 위치를 현재 시점에 맞추어 갱
신한다.
26:    END IF
27:    RETURN Sr
28: END IF

Algorithm End
```

(그림 4) 능동적 CBR 재조정을 위한 탐색 알고리즘.

의 ID들이 질의 처리 결과로 출력된다. 아래 (그림 4)는 이런 미래 시간 위치 질의를 처리하기 위한 탐색 알고리즘을 나타낸다.

알고리즘은 루트 노드에서부터 하위 노드로 내려가며 질의에 해당하는 노드를 탐색하며 내려간다. 특정 내부 노드의 엔트리가 질의에 해당하는 경우, 단계 4에서 재귀 함수 Search를 호출하여 자식 노드에 대하여 탐색해 내려간다. 이 자식 노드가 질의에 포함되어 능동적 재조정이 일어났을 경우, 단계 7~9에서 자식 노드의 변경을 고려하여 현재 엔트리의 CBR을 재조정 한다.

질의가 내부 노드에서 탐색해 내려와 단말 노드에 접근하였을 경우, 질의에 해당하는 이동 객체를 검색하기 위하여 단계 15~18을 실행한다. 그 후 단계 20 이하에서 제안한 기법에 따라 능동적 CAB 재조정의 수행 여부를 판단하며, 이익이 있다고 판단될 시 단계 25에서 리프 노드를 갱신한다. 알고리즘을 통하여 질의에 해당하는 이동 객체를 찾아냈을 때, 이동 객체의 ID는 S_r에 저장되며 알고리즘이 종료된 후 검색 결과로 반환된다.

4. 성능 비교 실험

본 장에서는 제안한 능동적 CBR 재조정 기법을 적용한 TPR*-트리와 기존 TPR*-트리의 미래 위치 질의에 대한 성능 비교를 통해 제안한 기법의 우수성을 보인다.

4.1 실험 환경

실험에 사용된 데이터 집합은 GSTD[The99]를 사용하여 생성하였다. GSTD는 이동 객체를 다룬 여러 논문들[Pfo00][Lin04]에서 사용한 데이터 생성기로서 다양한 특성의 데이터들을 생성할 수 있다. GSTD를 이용하여 10,000×10,000의 크기로 정규화 된 이차원 데이터 공간상에서 임의의 속도를 갖도록 생성된 100,000개의 점(point) 이동 객체들을 실험을 위한 데이터 집합으로 사용하였다. 이동 객체의 초기 분포는 GSTD에서 지원하는 균등 분포(uniform distribution), 스큐 분포(skewed distribution), 가우시안 분포(Gaussian distribution)의 세 가지 분포를 가지도록 설정하여 각각에 대하여 실험하였다.

이 이외에도 다양한 실험을 통해 정확한 성능을 측정하기 위하여 많은 매개 변수를 두었다. 밑의 <표 1>은 실험에 사용한 이동 객체의 속도, 갱신 주기, 질의 빈도, 질의 영역의 크기, 이동 객체의 속도, 그리고 예측 시점 등의 매개 변수들과 그 값을 정리한 것이다. 예를 들어, 이동 객체는 평균적으로 20, 50, 100, 150의 시간 마다 위치 정보를 갱신하며, 질의 영역의 크기는 전체 실험 공간에 비교해 0.01%에서 2.56%까지 조정된다.

<표 1>에서 굵은 글씨체의 변수 값은 그 매개 변수를 대표하는 값으로 특정한 하나의 매개 변수 값을 변화시켜 가

〈표 1〉 실험 환경에 사용된 매개 변수

매개 변수	매개 변수 값
평균 갱신 주기 (이동 객체 당)	10, 50, 100, 150
질의 빈도 (단위 시간 1당)	20, 40, 60, 80, 100
질의 영역 크기	0.01%, 0.16%, 0.64%, 2.56%
이동 객체 평균 속도	30, 50, 70, 90
미래 예측 시점	20, 40, 60, 80, 100

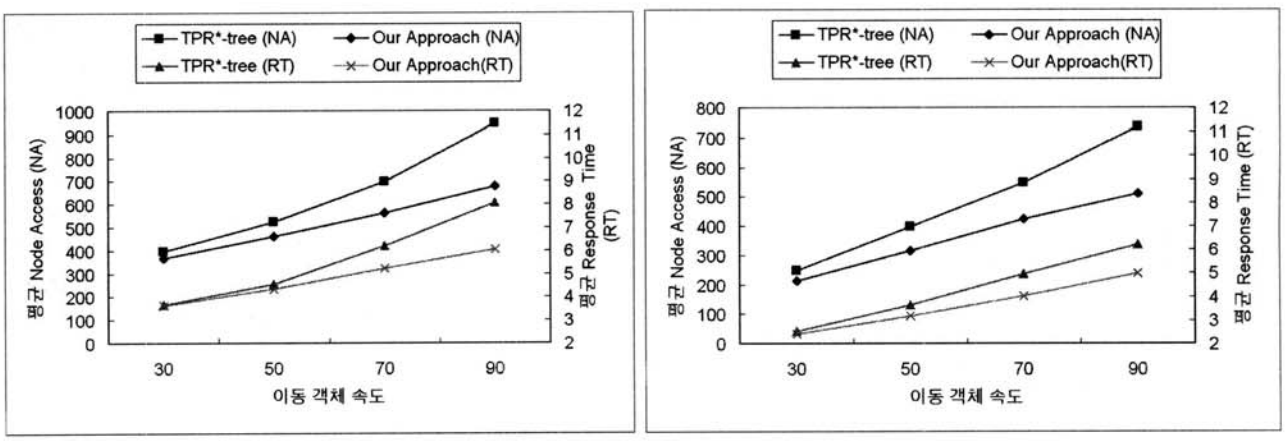
는 동안 다른 매개 변수 값들은 표에서 굵게 표시된 값으로 고정된다. 예를 들어, 갱신 주기를 20, 50, 100, 150으로 변화시키며 실험을 할 때, 질의 빈도, 질의 영역의 크기, 이동 객체 평균 속도, 그리고 미래 예측 시점은 각각 40, 0.16%, 50, 60으로 고정된다.

실험의 성능 척도로는 동일한 유형의 임의의 질의 100개의 처리에 소요된 노드 액세스 수와 질의 응답 시간의 평균값을 사용하였다. 노드 액세스 수 이외에 질의 응답 시간을 성능 척도로 둔 이유는 제안한 기법이 질의 처리에 필요한 노드 접근 회수를 줄여주면서, 추가적인 CPU 사용이나 메모리 사용으로 인한 성능 저하가 발생하지 않음을 확인하기 위해서이다. 모든 실험들은 윈도우즈 서버 2000에서 512MB의 메모리 크기와 2.23 GHz Pentium 4 CPU를 PC를 이용하여 수행되었다.

4.2 성능 비교

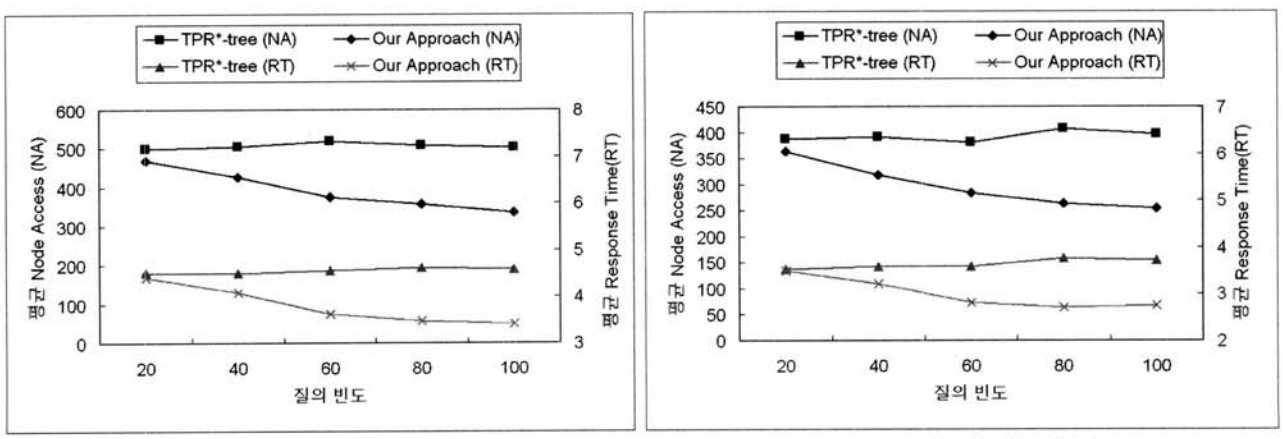
본 절에서는 제안하는 기법과 TPR*-트리의 성능 비교를 위한 실험들을 수행한다. 제안하는 기법의 정확한 성능 측정을 위하여 질의 처리 시 능동적 재조정 기법이 수행되면 이에 따르는 추가적인 디스크 액세스를 실험 결과에 반영한다. 즉, 추가적인 갱신 비용까지 고려한 인덱스의 전체적인 성능 향상 효과 정도를 측정하는 것이다. (그림 5)는 균등, 스큐 분포에 대해 이동 객체의 속도를 변화시켜가며 측정한 평균 노드 접근수와 평균 질의 응답 시간의 변화를 나타낸다. 참고로, 본 논문에서 가우시안 분포에 대한 실험 결과는 균등, 스큐 분포와 매우 유사하게 나타났으므로 지면 관계상 생략하였다.

(그림 5)에서 그래프 왼쪽의 세로축은 평균 노드 액세스 수, 오른쪽의 세로축은 평균 질의 응답 시간을 나타내고, 가로축은 매개 변수인 이동 객체 속도의 변화를 나타낸다. 이는 뒤에 나올 모든 실험 결과에서 동일하다. 실험 결과, 이동 객체의 속도가 점차 증가하는 상황에서도 제안하는 기법의 성능이 우월한 경향을 보였다. 평균 노드 액세스 수의 경우 균등 분포에서는 최대 29%, 스큐 분포에서는 최대 31%의 성능 향상 효과를 보였다. 평균 질의 응답 시간의 경우 균등 분포와 스큐 분포에 대해 최대 26%와 20%의 성능



(a) 균등 분포에 대한 성능 평가치 (b) 스큐 분포에 대한 성능 평가치

(그림 5) 이동 객체의 속도 변화에 대한 실험 결과



(a) 균등 분포에 대한 성능 평가치 (b) 스큐 분포에 대한 성능 평가치

(그림 6) 질의 빈도수의 변화에 대한 실험 결과

향상 효과를 보였다.

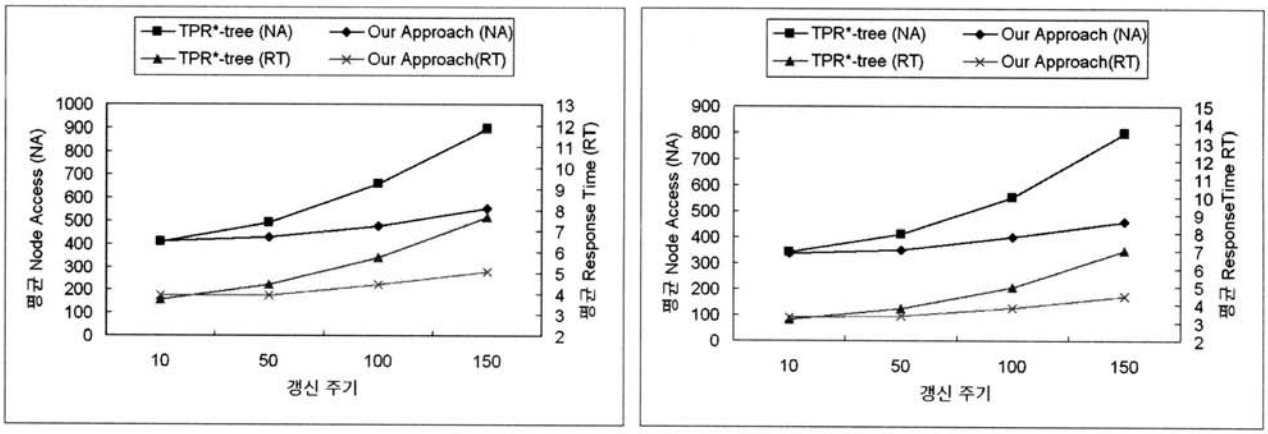
이렇게 이동 객체의 속도가 클수록 성능 향상 효과가 더 커지는 이유는 이동 객체의 속도에 맞춰 BR의 확장 속도도 빠르게 커져 사장 영역이 크게 증가하기 때문이다. 제안하는 기법은 기존 기법에 비해 자주 CBR 재조정을 수행할 수 있으므로 그만큼 성능 향상 효과가 크다. 평균 질의 응답 시간의 성능 향상 효과가 평균 노드 액세스 수의 성능 향상 효과와 동일하지 않은 이유는 제안하는 기법이 CAB 계산을 위해 CPU 시간을 추가적으로 사용하기 때문이다. 그럼에도 불구하고 전체적인 성능 향상 효과에는 큰 영향을 주지 않는다.

(그림 6)은 단위 시간 당 질의 빈도를 변화시켜 가며 성능을 측정한 결과이다. 실험 결과, 사용자 질의 빈도가 점차 증가하는 상황에서도 제안하는 기법의 성능이 우월한 경향을 보였다. 평균 노드 액세스 수의 경우 균등 분포에서는 최대 34%, 스큐 분포에서는 최대 37%의 성능 향상 효과를 보였다. 평균 질의 응답 시간의 경우 균등 분포의 경우 최대 26%, 스큐 분포의 경우 최대 28%의 성능 향상 효과를 보였다.

(그림 6)의 결과에서 알 수 있듯이 제안하는 방법은 시스템에서 처리할 질의가 증가하는 상황에서 더욱 큰 성능 개선 효과를 보인다. 이 실험의 또 다른 특징은 질의 빈도가 80이 넘는 상황에서는 성능 향상 비율이 일정하게 유지된다는 점이다. 이것은 능동적 CBR 재조정의 발생 빈도가 일정 크기의 질의 빈도를 넘는 상황에서는 거의 고정된다는 의미이다. 그 이유는 CAB 계산에 의하여 성능 향상 효과가 없다고 판단되는 경우에 관해서는 CBR 재조정을 수행하지 않기 때문이다.

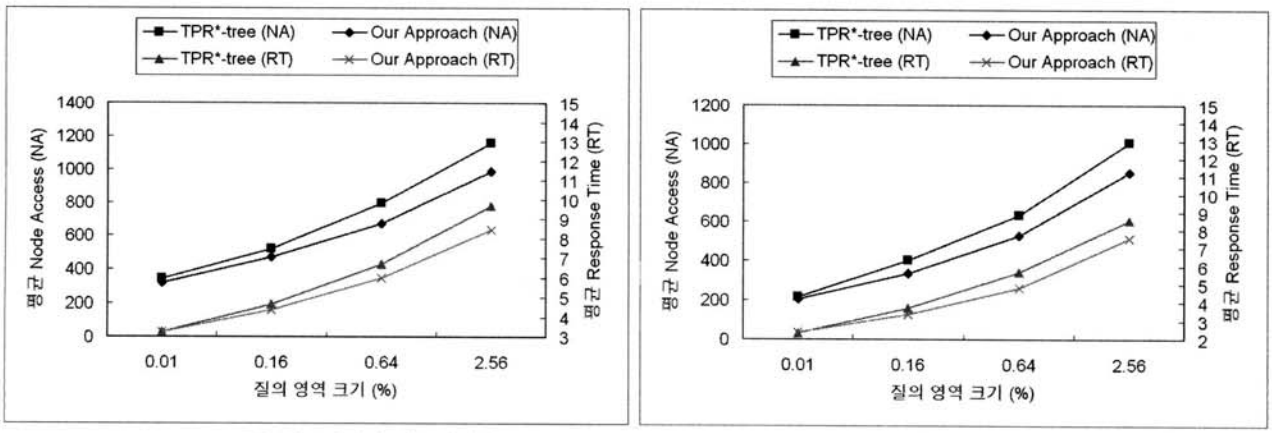
(그림 7)은 이동 객체 당 갱신 주기에 따른 성능 변화를 측정한 것이다. 그림에서 보듯이 이동 객체의 갱신 주기가 점차 커질수록 제안하는 기법의 성능 향상 효과도 크게 증가하였다. 평균 노드 액세스 수의 경우 균등 분포에서는 최대 39%, 스큐 분포에서는 43%의 성능 향상 효과를 보였다. 평균 질의 응답 시간의 경우 균등 분포에서 최대 34%, 스큐 분포에서 최대 37%의 성능 향상 효과를 보였다.

기존의 기법은 갱신 주기라는 제한된 상황에만 의지해 CBR 재조정을 해주기 때문에 이동 객체의 갱신 주기가 길



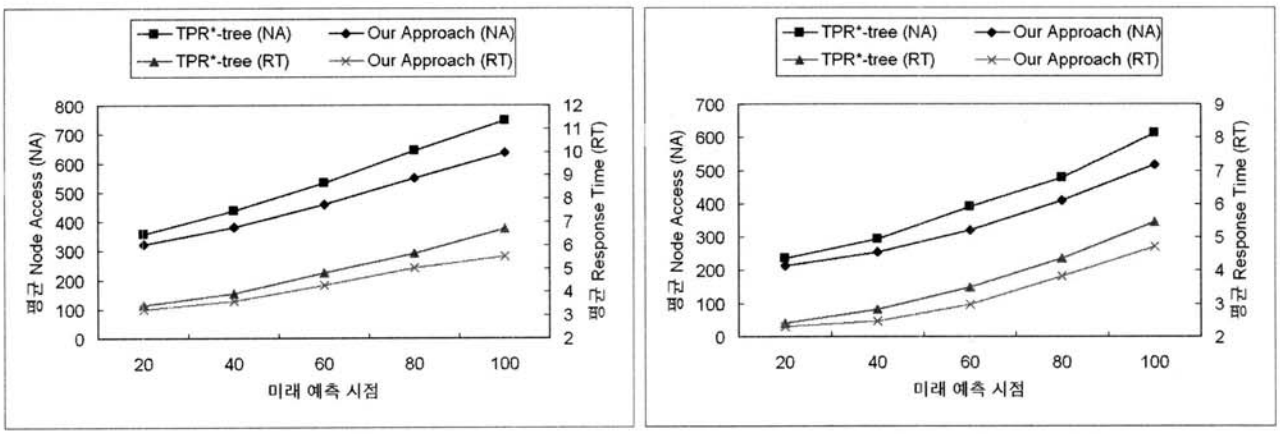
(a) 균등 분포에 대한 성능 평가 (b) 스큐 분포에 대한 성능 평가치

(그림 7) 갱신 주기의 변화에 대한 실험 결과



(a) 균등 분포에 대한 성능 평가치 (b) 스큐 분포에 대한 성능 평가치

(그림 8) 질의 영역의 크기 변화에 대한 실험 결과



(a) 균등 분포에 대한 성능 평가 (b) 스큐 분포에 대한 성능 평가치

(그림 9) 미래 예측 시점의 변화에 대한 실험 결과

면 그만큼 CBR 재조정이 일어나지 않았다. 그러나 제안하는 기법은 질의 시점을 이용하여 CBR 재조정 판단을 내리기 때문에 큰 성능 향상 효과를 가져 올수 있다. 갱신 주기가 10일 때와 같이 갱신이 지나치게 빈번하게 일어나는 경우는 CBR 재조정을 수행할 기회가 거의 없기 때문에 제안하는 기법이 기존의 기법과 비슷한 성능을 보인다. 하지만 일반적인 경우 갱신 주기가 이렇게 극단적으로 짧지 않기 때문에 제안하는 기법이 훨씬 좋은 성능 향상 효과를 보인다.

(그림 8)은 질의 영역 크기에 따른 성능 변화를 앞의 실험과 같은 방법으로 보인 것이다. 그림에서 보듯이 평균 노드 액세스 수의 경우 균등 분포와 스큐 분포에서 최대 17% 정도의 성능 향상 효과를 보였고, 평균 질의 응답 시간의 경우는 균등 분포와 스큐 분포에서 각각 최대 14%, 13%의 성능 향상 효과를 보였다. 질의 영역의 크기 변화는 다른 매개 변수들과는 달리 성능 향상에 많은 영향을 끼치지 않는 것으로 나타났지다. 하지만 질의 영역이 커지면 접근하는 단말 노드의 수도 증가하기 때문에 기본적인 성능 향상 효과를 얻을 수 있다.

(그림 9)는 질의에서 예측하고자 하는 시점의 변화에 따른 성능 비교 결과를 보인다. 실험 결과, 평균 노드 액세스 수의 경우 균등 분포에서 최대 15%, 스큐 분포에서 최대 16%의 성능 향상 효과를 보였다. 평균 질의 응답 시간의 경우 균등 분포에서 18%, 스큐 분포에서 16%의 성능 향상 효과를 보였다. 앞의 그림 8의 실험 결과와 유사하게 질의 예측 시점이 성능 향상에 많은 영향을 끼치지 않는 것으로 나타났다. 하지만 질의 예측 시점이 멀수록 접근되는 노드 수도 증가하기 때문에 기본적인 성능 향상 효과를 얻을 수 있다.

5. 결 론

본 논문에서는 TPR*-트리의 질의 처리 성능을 개선시키기 위한 능동적 재조정 기법에 관하여 제안하였으며, 다양한 실험을 통하여 제안하는 기법의 우수성을 증명하였다.

TPR*-트리는 이동 객체의 위치 인덱싱 및 미래 시점의 위치 질의의 처리를 위해 고안된 인덱스로서 인덱스 구성 이후에 미래 예측 시점이 경과함에 따라 인접 노드 간의 영역 중복이 심화되고 사장 영역이 증가하는 문제점이 존재하였다. 이런 사장 영역을 줄이기 위하여 TPR*-트리에서는 이동 객체의 갱신 시점에 맞춰 해당하는 노드의 CBR을 재조정해주는 정책을 사용하였다. 그러나 이동 객체의 갱신이 일어나기 전까지는 사장 영역이 지속적으로 증가하기 때문에 질의 처리 성능 저하를 방지하는 데는 한계가 있다. 또한, 갱신이라는 제한된 상황을 이용해서만 CBR 재조정이 일어날 수 있기 때문에 사장 영역은 시간이 갈수록 증가하며 갱신이 드문 경우에는 더욱 심각한 상황을 초래한다.

본 논문에서는 사장 영역의 증가라는 문제점을 해결하기 위하여 질의 처리 시에 CBR 재조정을 수행하는 새로운 기법을 제안하였다. 제안한 기법은 질의를 처리하면서, 루트에서 단말 노드로의 탐색 경로의 CBR 정보가 메모리 버퍼에 저장된다는 점에 착안, 이 정보를 이용하여 능동적인 방식으로 CBR을 재조정 한다. 이러한 능동적 방식의 CBR 재조정을 수행하기 위해서 필요한 비용과 재조정을 통해 이후의 질의 처리에서 얻을 수 있는 비용의 절감을 모두 고려하여 CBR 재조정의 필요 여부를 판단하는 비용 모델을 제시하였다. 제안된 기법의 우수성을 입증하기 위해 다양한 실험을 통하여 기존 TPR*-트리와 성능 비교를 수행하였다. 실험 결과, 제안한 방식은 TPR*-트리를 이용한 기존의 방식과 비교하여 최대 40% 이상의 성능 개선 효과를 보이는 것으로 나타났다.

감사의 글

이 논문은 2007년도 정부재원(교육인적자원부 학술연구조성사업비)으로 한국학술진흥재단의 지원(KRF-2007-314-D00221)과 지식경제부 및 정보통신연구진흥원의 대학 IT연구센터 지원(IITA-2008-C1090-0801-0040)의 연구결과임.

참 고 문 헌

[1] N. Beckmann, H. Kriegel, R. Schneider, and B. Seeger, "The R*-tree: An Efficient and Robust Access Method for Points and Rectangles," In *Proc. ACM Int'l. Conf. on Management of Data(ACM SIGMOD)*, pp.322-331, 1990.

[2] D. Comer, "The Ubuquitous B-tree," *ACM Computing Surveys*, Vol.11, pp.121-137

[3] D. L. Lee, J. Xu, B. Zheng, and W. C. Lee, "Data Management in Location-Dependent Information Services," *IEEE Pervasive Computing*, Vol.1, No.3, pp.65-72, 2002.

[4] B. Lin and J. Su, "On Bulk Loading TPR-Tree," In *Proc. IEEE Int'l. Conf. on Mobile Data Management*, pp.395-406, 2004.

[5] M. F. Mokbel, T. M. Ghanem, and W. G. Aref, "Spatio-Temporal Access Methods," *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering*, Vol. 26, No.2, pp.40-49, 2003.

[6] S. Prabhakar, Y. Xia, D. V. Kalashnikov, W. G. Aref, and S. E. Hambrusch, "Query Indexing and Velocity Constrained Indexing: Scalable Techniques for Continuous Queries on Moving Objects," *IEEE Trans. on Computers*, Vol.51, No.10, pp.1124-1140, 2002.

[7] D. Pfoser, C. S. Jensen, and Y. Theodoridis, "Novel Approaches in Query Processing for Moving Objects," In *Proc. Int'l. Conf. on Very Large Data Bases(VLDB)*, pp. 395-406, 2000.

[8] S. Saltenis, C. S. Jensen, S. T. Leutenegger, and M. A. Lopez, "Indexing the Positions of Continuously Moving Objects," In *Proc. ACM Int'l. Conf. on Management of Data (ACM SIGMOD)*, pp.331-342, 2000.

[9] A. P. Sistla, O. Wolfson, S. Chamberlain, and S. Dao, "Modeling and Querying Moving Objects," In *Proc. IEEE Int'l. Conf. on Data Engineering(IEEE ICDE)*, pp.422-432, 1997.

[10] Y. Tao, D. Papadias, and J. Sun, "The TPR*-Tree: An Optimized Spatio-Temporal Access Method for Predictive Queries," In *Proc. Int'l. Conf. on Very Large Data Bases (VLDB)*, pp.790-801, 2003.

[11] Y. Theodoridis, M. Vazirgiannis, and T. Sellis, "Spatio-Temporal Indexing for Large Multimedia Applications," In *Proc. IEEE Int'l. Conf. on Multimedia Computing and Systems(IEEE ICMCS)*, pp.441-448, 1996.

[12] Y. Theodoridis, R. Silva, and M. Nascimento, "On the Generation of Spatiotemporal Datasets," In *Proc. Int'l. Symp. on Spatial Databases*, pp.147-164, 1999.

[13] O. Wolfson, B. Xu, S. Chamberlain, and L. Jiang, "Moving Objects Databases: Issues and Solutions," In *Proc. Int'l. Conf. on Scientific and Statistical Database Management (SSDBM)*, pp.111-122, 1998.

[14] X. Xu, J. Han, and W. Lu, "RT-Tree: An Improved R-Tree Indexing Structure for Temporal Spatial Databases," In *Proc. Int'l. Symp. on Spatial Data Handling(SDH)*, pp.1040-1049, 1990.

김 상 욱

e-mail : wook@hanyang.ac.kr

1989년 2월 서울대학교 컴퓨터공학과 (학사)

1991년 2월 한국과학기술원 전산학과 (석사)

1994년 2월 한국과학기술원 전산학과 (박사)

1991년 7월~8월 미국 Stanford University, Computer Science Department 방문 연구원

1994년 2월-1995년 2월 KAIST 정보전자연구소 전문 연구원

1999년 8월~2000년 8월 미국 IBM T.J. Watson Research Center Post-Doc.

1995년 3월~2000년 8월 강원대학교 컴퓨터정보통신공학부 부교수

2003년 3월~현 재 한양대학교 정보통신대학 정보통신학부 교수

관심분야 : 데이터베이스 시스템, 저장 시스템, 트랜잭션 관리, 데이터 마이닝, 멀티미디어 정보 검색, 공간 데이터베이스/GIS, 주기억장치 데이터베이스, 이동 객체 데이터베이스/텔레매틱스, 사회 연결망 분석, 웹 데이터 분석

장 민 희

e-mail : zzmini@agape.hanyang.ac.kr

2003년 2월 홍익대학교 신소재공학과 (학사)

2006년 8월 한양대학교 정보통신학과 (석사)

2006년 9월~현 재 한양대학교 정보통신학과 박사과정

관심분야 : 데이터베이스 시스템, 데이터 마이닝, 공간 데이터베이스/GIS, 이동객체 데이터베이스, 사회 연결망 분석, 유사 쉐이더 검색

임 성 채

e-mail : sclim@dongduk.ac.kr

1992년 서울대학교 컴퓨터공학과(학사)

1994년 한국과학기술원 대학원 전산학과 (이학석사)

2004년 한국과학기술원 대학원 전산학과 (공학박사)

2000년~2000년 서울정보시스템 기술부 부장

2000년~2005년 2월 코리아와이즈넛 연구소 수석연구원 및 이사

2005년 3월~현 재 동덕여자대학교 컴퓨터전공 조교수

관심분야 : Web IR, 멀티미디어 시스템, 고성능 색인 기법