TPR*-트리의 성능 분석에 관한 연구

A Performance Study on the TPR*-Tree

김상욱* / Sang-Wook Kim
장민희** / Min-Hee Jang
임승환*** / Seung-Hwan Lim

요 약

TPR*-트리는 효과적으로 이동 객체의 미래 위치 예측을 수행하기 위하여 가장 널리 사용되는 인덱스 구조이다. 그러나 TPR*-트리는 인덱스 생성 이후 미래 예측 시점이 증가함에 따라 사장 영역과 영역 중복의 문제가 커지며, 이로 인하여 질의 처리 시 액세스되는 TPR*-트리 노드들의 수가 많아지는 성능 문제가 발생한다. 본 논문에서는 실험을 통하여 이러한 성능 저하의 문제점을 정량적으로 규명한다. 먼저, 미래 예측 시점이 증가함에 따라 질의 처리 성능이 얼마나 저하되는가를 보이고, 이동 객체의 위치 갱신 연산이 이러한 성능 저하 문제를 얼마나 완화시키는가를 보인다. 이러한 공헌은 TPR*-트리의 추가적인 성능 개선을 위한 정책을 고안하는데, 중요한 실마리를 제공할 수 있을 것이다.

Abstract

The TPR*-tree is the most widely-used index structure for effectively predicting the future positions of moving objects. The TPR*-tree, however, has the problem that both of the dead space in a bounding region and the overlap among bounding regions become larger as the prediction time in the future gets farther. This makes more nodes within the TPR*-tree accessed in query processing time, which incurs the performance degradation. In this paper, we examine the performance problem quantitatively with a series of experiments. First, we show how the performance deteriorates as a prediction time gets farther, and also show how the updates of positions of moving objects alleviates this problem. Our contribution would help provide important clues to devise strategies improving the performance of TPR*-trees further.

주요어 : 이동 객체, 미래 예측, TPR*-트리, 성능 개선
Keyword : moving objects, future prediction, TPR*-trees, performance improvement

■ 본 연구는 제주대학교를 통한 정보통신부 및 정보통신진흥원의 대학 IT연구센터 지원사업의 부분적인 지원을 받았음 (IITA-2005-C1090-0502-0009)
■ 논문접수 : 2005. 8. 27 ■ 심사완료 : 2005. 11. 22
* 교신저자 한양대학교 정보통신대학 정보통신학부 교수(wook@hanyang.ac.kr)
** 한양대학교 정보통신공학과 석사과정(zzmini@ihanyang.ac.kr)
*** 한양대학교 정보통신공학과 박사과정(firemoon@ihanyang.ac.kr)

1. 서 론

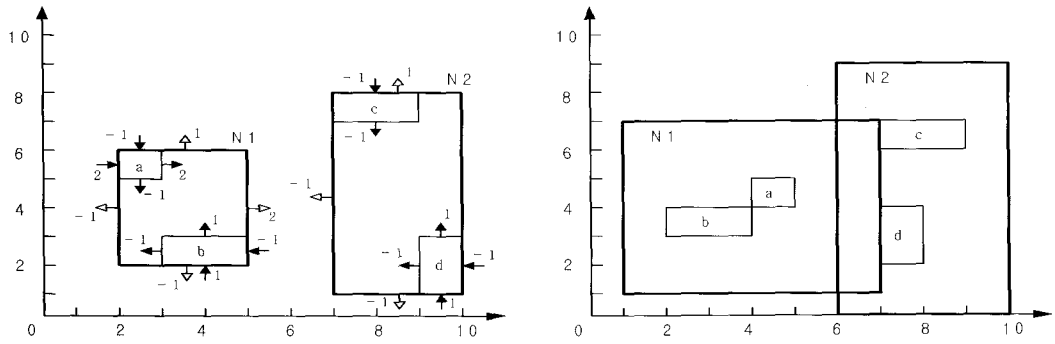
최근 들어, 이동 통신 및 GPS(global positioning system) 기술의 발달로 인해 객체의 이동 정보를 효과적으로 활용하기 위한 방안에 대한 관심이 증대되고 있다. 이동 객체는 주기적으로 자신의 위치를 서버로 전송하는데, 이 데이터들은 시간의 흐름에 따라 공간적인 위치 정보가 변화하는 시공간 데이터(spatio-temporal data)의 특성을 갖는다[Lee02]. 이동 객체 데이터베이스는 이와 같이 시간의 흐름에 따른 객체의 위치 변화 정보가 저장된 데이터베이스를 의미한다[Wol98].

이동 객체 데이터베이스에 대한 사용자의 질의는 이동 객체의 과거의 움직임을 검색하는 과거 시간 질의(past-time query)[Xu90][The96]와 이동 객체의 미래의 움직임을 예측하여 검색하는 미래 시간 질의(future-time query)로 나눌 수 있다[Sis97]. 대표적인 미래 시간 질의의 예로는 "1:00 pm에 한남 대교를 지나는 모든 차량들을 검색하라" 를 들 수 있다. 본 논문에서는 미래 시간 질의를 효과적으로 수행하기 위한 방안에 대하여 논의한다.

대용량의 이동 객체 데이터베이스에서 사용자 질의의 빠른 처리를 보장하기 위해서는 효과적인 인덱싱 방안이 수반되어야 한다[Mok03]. 미래 시간 질의를 처리하기 위한 인덱스 구조로는 VCI-트리[Pra02], TPR-트리[Sal00], TPR*-트리[Tao03] 등이 제안되었으며, 이들 중 TPR-트리의 단점을 개선한 TPR*-트리가 가장 널리 사용되고 있다. TPR*-트리는 객체들이 공간상의 각 축에서 갖는 최대, 최소 속도 값을 저장하여, 이를 이동 객체의 미래 위치를 검색하기 위한 정보로 이용한다. TPR*-트리는 인덱스의 생성 이후에 미래 예측 시점이 증가함에 따라 각 노드에 대응하는 영역간의 중복이 심화되고, 사장 영역(dead space)[Bec90]이 증가한다는 문제점을 갖고 있다. 이 결과, 트리 검색 시에 액세스되는 노드들의 수가 많아지며, 이는 질의 처리의 급격한 성능 저하를 초래한다[Tao03].

본 논문에서는 다양한 요인들이 TPR*-트리의 성능에 미치는 영향을 실험적으로 검증한다. 우선, TPR*-트리의 인덱스 생성 이후에 미래 예측 시점이 증가함에 따라서 질의 처리의 성능이 어떻게 저하되는지를 정량적으로 보이고, TPR*-트리의 갱신 연산이 이러한 성능 저하를 얼마만큼 완화시키는지를 분석한다. 이들 결과를 종합하여 TPR*-트리의 갱신과 질의 처리 성능의 상관 관계를 규명한다. 이러한 TPR*-트리의 갱신 연산이 질의 처리에 미치는 영향에 대한 정량적 실험 결과의 제시는 이전 연구에서는 수행되지 않은 것으로서, 현재 가장 널리 사용되는 TPR*-트리의 추가적인 성능 개선에 필요한 정책을 고안하는 데에 중요한 실마리를 제공할 수 있을 것이다.

본 논문의 구성은 다음과 같다. 제 2 장에서는 관련 연구로서 기존의 미래 예측을 위한 인덱스 구조들을 소개하고, 그 장단점을 지적한다. 제 3 장에서는 본 논문의 연구 동기로서 TPR*-트리의 성능 저하 문제의 근본적인 원인에 대하여 논의한다. 제 4 장에서는 미래 예측 시점의 증가에 따른 TPR*-트리의 성능 저하 문제와 TPR*-트리의 갱신과 성능간의 상관 관계를 다양한 실험을 통하여 정량적으로 분석한다. 마지막으로, 제 5 장에서는 본 논문을 요약하고 결론을 내린다.



(a) time 0. (b) time 1.

<그림 1> TPR-트리에서 미래 예측을 위한 CBR의 확장.

2. 관련 연구

본 장에서는 관련 연구로서 미래 예측을 위한 인덱스 구조들을 소개하고, 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[Sal00]. 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 객체들의 각축에 대한 최대, 최소의 속도로 구성된다. 미래의 임의의 시점에서 이동 객체의 위치는 CBR에 포함된 이동 객체의 위치와 속도 정보를 이용하여 계산을 통해 예측할 수 있다. CBR 내의 MBR의 현재 위치와 각 축의 최대, 최소속도를 이용하여 계산되어 예측된 노드의 영역을 BR(bounding rectangle)이라 정의 한다.

그림 1은 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. 이 예에서는 이동 객체의 각축에서의 최대, 최소 이동 속도가 동일하다고 가정한다. 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 MBR의 이동을 고려하여 확장된 time 1 시점의 BR을 나타낸다.

TPR-트리의 CBR은 내부의 이동 객체들의 위치와는 상관없이 단순히 최대, 최소 속도만을 가진다. 따라서 그림 1에서 보는 바와 같이 해당 CBR에 대한 BR은 시간의 흐름에 따라 단순 확장하는 특성을 보인다. TPR-트리에서는 BR이 무한대로 확장하는 것을 방지하기 위해서 이동 객체의 위치 정

보 갱신 시에 객체의 실제 위치를 반영하여 MBR을 재계산하는 정책을 사용한다.

2.2 TPR*-트리

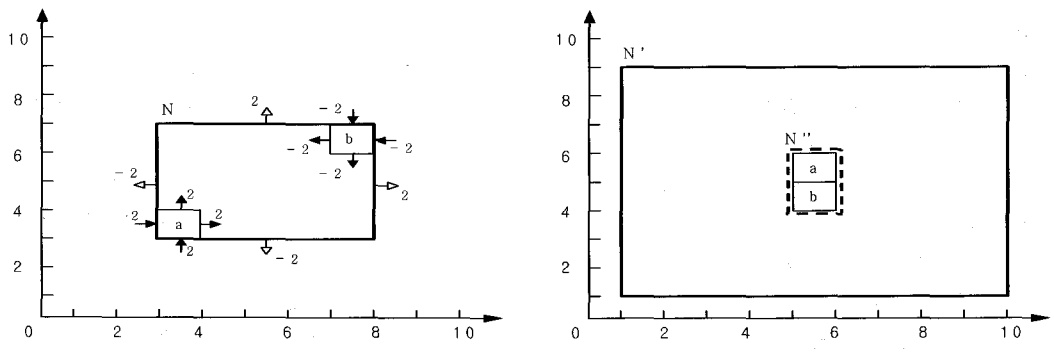
TPR*-트리[Tao03]는 기본적으로 TPR-트리와 동일한 구조를 사용한다. 그러나 갱신 연산 시에 TPR-트리가 R*-트리의 삽입, 삭제 알고리즘을 그대로 사용하는데 비하여 TPR*-트리에서는 객체의 이동성을 고려한 향상된 알고리즘을 사용한다. 이로 인해 TPR*-트리는 TPR-트리에 비하여 갱신 및 검색 성능을 개선할 수 있다. TPR-트리는 단지 이동 객체 갱신 시점에서만의 MBR의 면적, 둘레, 겹침, 거리를 고려하기 때문에 객체가 시간에 따라 이동한다는 특성은 반영하지 못한다. 그러나 TPR*-트리는 이동 객체 갱신 시점 이후의 시간에 따라 노드에 대응하는 BR이 확장하는 영역(sweeping region)의 면적을 최소화하는 방식으로 갱신을 수행한다[Tao03].

예를 들어, TPR-트리는 이동 객체가 삽입된 시점에서 노드에 대응되는 MBR의 면적 확장을 최소로 가지는 노드를 찾아 이동 객체를 삽입한다. 반면, TPR*-트리의 삽입 알고리즘은 삽입 시점 이후에 확장하는 BR을 최소로 가지는 노드를 찾아 이동 객체를 삽입한다. 루트 노드에서부터 하위 노드로 내려가며 이동 객체가 삽입되는 영역에 대응되는 노드들이 삽입 시점 이후에 확장하는 영역을 계산하여 우선 순위 큐에 저장한다. 그리고 그 값이 가장 작은 노드에 삽입함으로서 이동 객체에 맞는 최적의 노드를 찾아낸다.

이러한 전략으로 인하여 TPR*-트리는 TPR-트리의 갱신 알고리즘보다 많은 갱신 비용을 요구한다. 그러나 인덱스 구조를 보다 잘 구성하기 때문에 전체적인 질의 처리 성능을 크게 개선할 수 있다. TPR*-트리의 실험 결과에 의하면 TPR*-트리는 TPR-트리에 비해 최소 1.5배에서 최대 5배까지의 성능 개선을 보였다[Tao03].

3. 연구 동기

이 장에서는 미래 예측 시점이 증가함에 따라 TPR*-트리의 질의 처리 성능이 저하되는 근본적인 이유에 대하여 논의한다. 또한, 이동 객체의 위치 정보 갱신 연산이 질의 처리 성능에 영향을 미치는 이유에 대하여 분석한다.



(a) time 0. (b) time 1.

<그림 2> BR의 확장

3.1. 성능 저하 문제

앞장에서 언급한 바와 같이, TPR*-트리의 CBR은 포함된 객체들이 공간상의 각 축에서 갖는 속도들 중 최대, 최소값만을 저장하기 때문에 각 노드의 BR은 지속적으로 확장 된다[1]. 이로 인하여, 시간이 흐를수록 하나의 노드에 대응하는 BR의 면적이 증가하게 되어서 노드간의 영역 중복이 심화되고, 사장 영역이 증가된다. 이 결과, 노드 검색을 위한 디스크 액세스 수가 증가 되어 질의 처리 성능이 떨어지게 된다.

그림 2는 time 0 시점에 하나의 노드와 대응하는 CBR이 이후의 time 1 시점에 BR로 확장되는 것을 나타낸다. 그림 2(a)의 객체 a, b는 각각의 속도 값에 의해서 time 1 시점에서 그림 2(b)와 같이 위치가 변화한다. 반면, TPR*-트리의 CBR은 객체들이 공간상의 각 축에서 갖는 최대, 최소 속도 값만을 가진다. 그러므로 time 1 시점에서 실제 이동 객체의 위치는 N''으로 이동하였음에도 불구하고 MBR N은 time 1 시점에서 그림 2(b)의 N'과 같이 크게 확장 된다. 이와 같은 BR의 불필요한 확장은 큰 사장 영역을 유발 시킬 뿐만 아니라 다른 BR과의 영역 중복을 발생시킨다. 이들은 모두 질의 처리시의 성능 저하를 초래시키는 부정적인 요인들이다. 제 4장에서는 BR의 불필요한 확장으로 인한 TPR*-트리의 성능 저하 현상을 다양한 실험을 통하여 정량적으로 검증한다.

3.2. 갱신 연산

TPR*-트리에서는 최대, 최소 속도를 갖는 CBR이 무한대로 확장하는 것을 방지하기 위해서 이동 객체의 위치 정보 갱신 시에 객체의 실제 위치를 반영하여 MBR을 재계산하는 정책을 사용한다. 그림 2(b)에서 time 1 시점에서의 객체 a, b를 포함하는 BR N'은 많은 사장 영역을 유발하였으나, time 1 시점에서 객체의 위치 정보를 갱신시킨다면, TPR*-트리에서는 BR 보정 알고리즘에 의해서 BR의 크기를 N'에서 N''으로 줄일 수 있는 기회가 될 수 있다. 사장 영역과 중복 영역의 크기가 감소하게 되므로 전체 질의 처리 성능은 개선된다. 제 4장에서는 갱신 연산과 TPR*-트리의 성능 개선의 상관 관계를 실험적으로 분석한다.

4. 실 험

본 장에서는 실험에 의한 성능 분석을 통하여 미래 예측 시점의 증가에 따른 TPR*-트리의 성능 저하와 TPR*-트리의 갱신 연산이 성능에 미치는 영향을 정량적으로 검증한다. 먼저, 성능 평가를 위한 실험 환경을 설명하고, 다양한 실험을 통하여 TPR*-트리의 성능 변화를 관찰하고 이를 분석한다.

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. 10,000 X 10,000의 크기로 정규화된 이차원 데이터 공간상에서 [0, 50] 사이의 임의의 속도를 갖도록 생성된 100,000개의 점(point) 객체들을 실험을 위한 데이터 집합으로 사용하였다. 이 객체들의 공간상의 위치 분포는 가우시안(gaussian) 분포를 가진다.

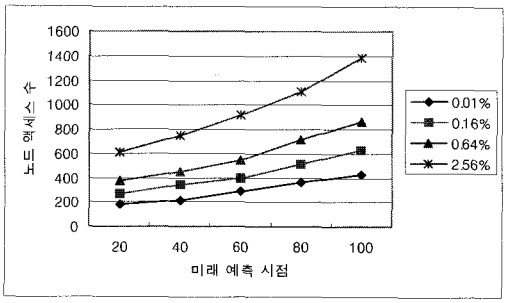
미래 시간 질의를 위한 질의 영역은 전체 데이터 공간의 0.01%, 0.16%, 0.64%, 2.56%의 면적을 갖도록 생성하였고, 질의 영역들의 위치 분포는 균등(uniform) 분포를 갖도록 설정 하였다. 생성된 질의는 질의 발

1. CBR이 내부의 모든 객체들의 이동을 대표하는 값으로서 각 축의 최대, 최소의 속도 값만을 저장하는 이유는 그 CBR의 하위 노드들에 저장된 모든 객체들에 대하여 위치와 속도 값을 유지함으로써 발생하는 지나친 저장 공간의 오버헤드를 피하기 위해서 이다.

생 시점을 기준으로 [10, 100] 내의 시점 이후의 미래 예측 시점을 갖도록 설정하였다. 성능의 척도로서 동일한 유형의 임의의 질의 100개의 처리에 소요된 노드 액세스 횟수의 평균을 사용하였고, 이를 미래 예측 시점을 달리하여 측정하였다.

본 논문에서는 미래 예측 시점의 변화에 따른 성능의 변화를 살펴보기 위하여 다음의 세 가지 실험을 수행한다. 실험 1에서는 질의 영역의 크기를 변경하며 미래 예측 시점에 따른 성능을 측정한다. 실험 2에서는 이동 객체 수를 변경하며 미래 예측 시점에 따른 성능을 측정한다. 실험 3에서는 동적인 환경에서 갱신 빈도를 변경하며 미래 예측 시점에 따른 성능을 측정한다.

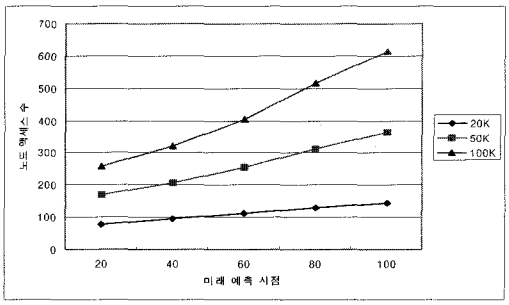
실험 1은 질의 영역의 크기를 변화시켜가며 미래 예측 시점에 따른 성능 변화를 살펴보기 위한 실험이다. 질의 영역의 크기는 전체 데이터 공간 면적의 0.01%, 0.16%, 0.64%, 2.56%로 하였고, 모든 질의는 TPR*-트리 생성 직후 시점인 0에 발생 시킨 후 미래 예측 시점을 20, 40, 60, 80, 100으로 변경하며 성능의 변화를 살펴보았다. 그림 3은 실험 1의 결과를 보인 것이다. 가로축은 미래 예측 시점, 세로축은 수행된 100개의 질의에 대한 평균 노드 액세스 수를 나타낸다.



<그림 3> 질의 영역의 크기 변경에 따른 성능 비교

실험 결과, 미래 예측 시점이 증가할수록 노드 액세스 수도 크게 증가하는 경향을 보였다. 질의 영역이 2.56%일 때, 미래 예측 시점이 20인 경우에 비하여, 미래 예측 시점이 40인 경우 24%, 미래 예측 시점이 60인 경우 50%, 미래 예측 시점이 80인 경우 86%, 미래 예측 시점이 100인 경우 126%의 추가적인 노드 액세스가 발생하는 것으로 나타났다. 질의 영역을 0.16%, 0.64%, 2.56%로 변화하여 실험하였을 때에도 노드 액세스는 비슷한 비율로 증가하는 결과를 보였다.

실험 2는 이동 객체 수를 변화시켜가며 미래 예측 시점에 따른 성능 변화를 살펴보기 위한 실험이다. 이동 객체의 개수를 20,000, 50,000, 100,000개로 달리하였고, 미래 예측 시점은 20, 40, 60, 80, 100으로 하였으며, 질의 영역의 크기는 0.16%로 설정하였다. 그림 4는 실험 2의 결과를 보인 것이다. 가로축은 미래 예측 시점, 세로축은 수행된 100개의 질의에 대한 평균 노드 액세스 수를 나타낸다.

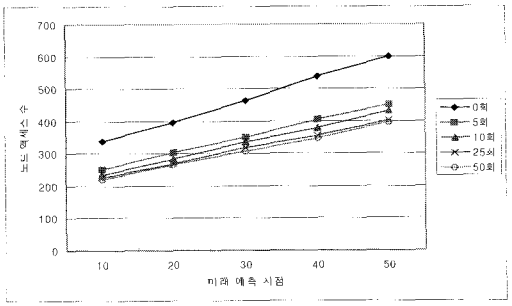


<그림 4> 이동 객체 개수의 변경에 따른 성능 비교

실험 결과, 모든 경우에서 미래 예측 시점이 증가할수록 노드 액세스 수도 크게 증가하는 경향을 보였다. 이동 객체가 20,000개일 때, 미래 예측 시점이 20인 경우에 비하여, 미래 예측 시점이 40인 경우 10%, 미래 예측 시점이 60인 경우 27%, 미래 예측 시점이 80인 경우 50%, 미래 예측 시점이

100인 경우 83%의 추가적인 노드 액세스가 발생하는 것으로 나타났다. 이동 객체의 개수를 늘려 50,000개, 100,000개로 변화하여 실험하였을 때에의 성능 저하는 더욱 심화되었다. 이동 객체가 50,000개일 때, 미래 예측 시점이 20일 때에 비하여, 미래 예측 시점이 40인 경우 16%, 미래 예측 시점이 60인 경우 42%, 미래 예측 시점이 80인 경우 75%, 미래 예측 시점이 100인 경우 114%의 추가적인 노드 액세스가 발생하는 것으로 나타났다. 이동 객체가 100,000개일 때, 미래 예측 시점이 20일 때에 비하여, 미래 예측 시점이 40인 경우 18%, 미래 예측 시점이 60인 경우 51%, 미래 예측 시점이 80인 경우 90%, 미래 예측 시점이 100인 경우 137%의 추가적인 노드 액세스가 발생하는 것으로 나타났다. 이는 이동 객체 데이터베이스의 크기가 커짐에 따라 미래 예측 시점의 증가에 따르는 성능 저하 현상이 더욱 심화된다는 것을 의미한다.

실험 3은 갱신 빈도를 변화시켜 가며 미래 예측 시점에 따른 성능 비교의 변화를 관찰한 실험이다. 이를 위해서 각 이동 객체의 갱신 빈도를 0회, 5회, 10회, 25회, 50회로 달리하면서 갱신을 수행하였다. 각 갱신 연산은 [0, 50] 사이의 임의의 시각에 수행되도록 하였다. 미래 시간 질의도 [0, 50] 사이의 발생하도록 함으로써 갱신과 질의의 수행이 병행되도록 실험하였다. 질의의 미래 예측 시점은 10, 20, 30, 40, 50으로 하였고, 질의 영역의 크기는 0.16%로 설정하였다. 질의가 발생된 시점에 의해 실험 결과가 달라지는 경우를 방지하기 위하여 질의가 들어오는 시점으로부터 10, 20, 30, 40, 50후의 상대적인 미래 시점을 설정한 후, 동일한 미래 예측 시점을 가지는 질의 처리 시간의 평균을 계산하였다. 그림 5는 실험 3의 결과를 보인 것이다. 가로축은 미래 예측 시점, 세로축은 수행된 100개의 질의에 대한 평균 노드 액세스 수를 나타낸다.



<그림 5> 갱신 빈도의 변경에 따른 성능 비교

미래 예측 시점이 50일 때, 갱신 연산을 수행하지 않은 경우를 기준으로 5회 갱신을 수행한 경우 75%, 10회 수행한 경우에 비하여 73%, 25회 수행한 경우에 비하여 69%, 50회 수행한 경우에 비하여 66% 정도로 노드 액세스 수가 감소하였다. 이는 빈번한 갱신 연산이 미래 시간 질의의 처리 성능에 긍정적인 영향을 준다는 것을 의미한다. 그 이유는 갱신 연산으로 인하여 미래 예측 시점에서의 BR의 크기가 감소하여 노드간의 영역 중복과 사장 영역을 줄이기 때문이다.

5. 결 론

본 논문에서는 TPR*-트리의 질의 처리 성능을 개선시키는 데에 중요하게 작용하는 요인을 분석하기 위하여 다양한 실험을 통하여 TPR*-트리의 질의 처리 성능을 정량적으로 검증하였다. TPR*-트리는 인덱스 구성 이후에, 미래 예측 시점이 경과함에 따라 인접 노드간의 영역 중복이 심화되고, 각 노드에 대응하는 BR의 사장 영역이 증가하는 문제점을 안고 있다. 이 문제를 정량적으로 파악하기 위해 본 논문에서는 미래 예측 시점이 경과함에 따른 TPR*-트리의 성능 저하 현상을 실험을 통하여 검증하였다. 또

한, TPR*-트리의 갱신 연산의 BR의 크기 보정 정책이 TPR*-트리의 질의 처리 성능에 어떠한 영향을 미치는지를 검증하였다.

실험 결과, 미래 예측 시점이 증가할수록 TPR*-트리의 질의 처리를 위한 노드 액세스 수도 크게 증가하는 경향을 보였다. 또한, 이동 객체 데이터베이스의 크기가 커짐에 따라 미래 예측 시점의 증가에 따르는 성능 저하 현상이 더욱 심화되는 것으로 나타났다. 한편, 빈번한 갱신 연산이 미래 예측 시점의 증가에 따른 TPR*-트리의 성능 저하를 크게 완화시키는 것으로 나타났다. 그 이유는 갱신 연산으로 인하여 미래 예측 시점에서의 BR의 크기가 감소하여 노드간의 영역 중복과 사장 영역을 줄이기 때문이다.

TPR*-트리의 성능에 영향을 미치는 여러 요인들과 성능의 상관 관계는 TPR*-트리의 성능 개선을 위한 정책의 결정에 중요한 요인으로 작용할 것이다. 이러한 연구 결과를 기반으로 TPR*-트리의 성능 저하 문제를 새로운 각도에서 개선하는 방법을 향후 연구로서 수행중이다.

참고 문헌

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. the ACM Int'l. Conf. on Management of Data(ACM SIGMOD)*, 1990, pp. 322-331.
2. D. L. Lee, J. Xu, B. Zheng, and W. C. Lee, "Data Management in Location-Dependent Information Services," In *Proc. IEEE Pervasive Computing*, Vol. 1, No. 3, 2002, pp. 65-72.
3. B. Lin and J. Su, "On Bulk Loading TPR-Tree," In *Proc. Proc. of IEEE Int'l. Conf. on Mobile Data Management*, 2004, pp. 395-406.
4. M. F. Mokbel, T. M. Ghanem, and W. G. Aref, "Spatio-Temporal Access Methods," In *Proc. Bulletin of the IEEE Computer Society Technical Committee on Data Engineering*, Vol. 26, No. 2, 2003, pp. 40-49.
5. 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," In *Proc. IEEE Trans. on Computers*, Vol. 51, No. 10, 2002, pp. 1124-1140.
6. D. Pfoser, C. S. Jensen, and Y. Theodoridis, "Novel Approaches in Query Processing for Moving Objects," In *Proc. the Int'l. Conf. on Very Large Data Bases(VLDB)*, 2000, pp. 395-406.
7. S. Saltenis, C. S. Jensen, S. T. Leutenegger, and M. A. Lopez, "Indexinng the Positions of Continuously Moving Objects," In *Proc. the ACM Int'l. Conf. on Management of Data(ACM SIGMOD)*, 2000, pp. 331-342.
8. A. P. Sistla, O. Wolfson, S. Chamberlain, and S. Dao, "Modeling and Querying Moving Objects," In *Proc. IEEE Conf. on Data Engineering(ICDE)*, 1997, pp. 422-432.
9. Y. Tao, D. Papadias, and J. Sun, "The TPR*-Tree: An Optimized Spatio-Temporal Access Method for Predictive Queries," In *Proc. the Int'l. Conf. on Very Large Data Bases(VLDB)*, 2003, pp. 790-801.
10. Y. Theodoridis, M. Vazirgiannis, and T. Sellis, "Spatio-Temporal Indexing for

Large Multimedia Applications," In *Proc. the IEEE Conf. on Multimedia Computing and Systems(ICMCS)*, 1996, pp. 441-448.

11. Y. Theodoridis, R. Silva, and M. Nascimento, "On the Generation of Spatiotemporal Datasets," In *Proc. the Int'l. Symp. on Spatial Databases*, 1999, pp. 147-164.
12. O. Wolfson, B. Xu, S. Chamberlain, and L. Jiang, "Moving Objects Databases: Issues and Solutions," In *Proc. the Int'l. Conf. on Scientific and Statistical Database Management(SSDBM)*, 1998, pp. 111-122.
13. X. Xu, J. Han, and W. Lu, "RT-Tree: An Improved R-Tree Indexing Structure for Temporal Spatial Databases," In *Proc. the Int'l. Symp. on Spatial Data Handling, SDH*, 1990, pp. 1040-1049.

김상욱

1989년 서울대학교 컴퓨터공학과 졸업(학사)
1991년 한국과학기술원 전산학과 졸업(석사)
1994년 한국과학기술원 전산학과 졸업(박사)
1991년 미국 Stanford University, Computer Science Department 방문 연구원
1994년~1995년 KAIST 정보전자연구소 전문 연구원
1999년 2000년 8월: 미국 IBM T.J. Watson Research Center Post-Doc.
1995년~2000년 강원대학교 컴퓨터정보통신공학부 부교수
2003년~현재 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

장민희

2003년 홍익대학교 신소재공학과 졸업(학사)
2006년 한양대학교 정보통신공학과 졸업 예정(석사)
관심분야: 데이터베이스 시스템, 저장 시스템, 데이터 마이닝, 공간 데이터베이스/GIS, 이동 객체 데이터베이스/텔레매틱스, 데이터베이스 보안

임승환

2003년 한양대학교 전자컴퓨터공학부 졸업(학사)
2005년 한양대학교 정보통신공학과 졸업(석사)
2005년 한양대학교 정보통신공학과 입학(박사)
관심분야: 데이터베이스 시스템, 저장 시스템, 데이터 마이닝, 사회 연결망 분석