영역 질의 처리를 위한 TB-tree의 삽입 알고리즘

장종우[O] 임덕성[+] 홍봉희
부산대학교 컴퓨터 공학과, 영진전문대학 컴퓨터정보기술계열[+]
{jwchang[O], bhhong}@pusan.ac.kr, junsung@yjc.ac.kr[+]

The Insertion Algorithm of TB-tree for Improving Range Queries

Jongwoo Chang[O] Duksung Lim[+] Bonghee Hong
Dept. of Computer Engineering, Pusan National University,
Division of Computer Information Technology, Yeungjin College[+]

요 약

차량과 같이 시간의 흐름에 따라 위치를 변경하는 객체를 이동체라 한다. 이동체의 과거 궤적은 시간이 지남에 따라 누적되므로 대용량 정보가 된다. 대용량 궤적 정보를 저장하는 이동체 데이터베이스에서 효율적으로 궤적을 검색하기 위해서는 색인이 필요하다. 특히 궤적을 선택하는 과정과 선택된 궤적의 일부분을 추출하는 과정으로 이루어진 복합 질의를 처리하기 위해서는 궤적 보존을 지원하는 TB-tree와 같은 색인 구조가 적합하다. 그러나 TB-tree와 같이 시간적으로 잘 구성된 색인은 공간적인 겹침이 커지는 문제가 있고, 반대로 공간적으로 잘 구성된 색인은 시간 도메인의 겹침을 심화시키는 문제점이 있다.

이 논문에서는 시간 도메인 중심의 분할 정책과 공간 도메인 중심의 분할 정책을 분석하여 서로 다른 두 도메인 사이의 관계를 밝힐 수 있는 파라미터를 제안하고, 이를 TB-tree에 적용하여 TB-tree의 장점을 유지하면서 영역 질의에 효과적인 분할 정책을 설계 및 구현한다. 또한 성능 평가를 통하여 제안된 분할 정책이 기존의 TB-tree 보다 영역 질의에서 우수함을 보인다.

1. 서 론

건물이나 지형과 같은 지리 객체와 달리 차량, 선박은 시간의 흐름에 따라 자신의 위치를 변경하는 이동체의 특성을 가진다. 최근 이러한 이동체와 관련하여 GPS(Global Positioning System) 기술 및 무선 통신 기술에 기반한 위치 기반 서비스(LBS : Location Based Service)에 대한 요구가 증가하고 있다. 위치 기반 서비스에서 효과적으로 이동체의 위치를 저장하고 검색하기 위해서는 이동체 데이터베이스가 필요하다.

이동체의 위치를 일정 시간 간격 또는 이동체의 속도나 방향의 변경이 있을 때 마다 저장할 경우 다수의 이동체로부터 획득된 궤적 정보의 양은 시간이 흐름에 따라 누적되므로 대용량의 정보가 된다. 이러한 궤적 데이터에 대하여 특정 시간 동안 주어진 장소에 있었던 이동체를 찾는 시공간 영역 질의와 영역 질의 등으로 선택된 이동체의 궤적을 추출하는 복합 질의를 효과적으로 처리할 필요가 있다.

이동체의 궤적을 색인할 수 있는 방법으로 시공간 색인인 3D R-tree[1]가 있다. 3D R-tree는 시간과 공간에 대한 색인을 분리하여 유지하지 않고 3차원 MBB(Minimum Bounding Box) 형태로 저장한다. 따라서 시간과 공간에 대한 색인이 각각 존재하는 것보다 3차원 영역 질의에 효과적이다. 그러나 R-tree[2]의 삽입 및 분할 정책은 연대순(chronology)으로 삽입되는 이동체의 특성을 고려하지 않아 공간 활용도가 저하되는 단점이 있다. 따라서 이동체가 많을 경우 색인의 크기가 크고, 동일한 이동체의 궤적이 많은 노드에 분산 저장되어 높은 궤적 추출 비용이 필요하므로 복합 질의에 효율적이지 못하다.

복합 질의를 효율적으로 처리할 수 있는 궤적 색인으로 TB-tree[3]가 있다. TB-tree는 공간 활용도가 높으므로 대용량의 궤적 정보를 저장하기에 적합하다. 그리

고 TB-tree는 궤적 추출을 효과적으로 하기 위하여 단말 노드에 동일한 이동체의 궤적을 모아두는 궤적 번들 기법을 사용한다. 또한 서로 다른 단말 노드에 저장된 동일한 궤적간에 연결 리스트가 존재하여 다시 비단말 노드를 접근하지 않고 다음 궤적이 있는 단말 노드로 이동할 수 있다. 그러나 궤적 번들 기법으로 인하여 R-tree와 같은 공간적인 특성을 고려하지 못하여 비단말 노드에서의 겹침(overlap)과 사장 영역(dead space)을 크게 만들어 영역 질의시 노드 방문 회수를 증가시키는 문제점이 있다.

이 논문에서는 효율적인 복합 질의와 영역 질의 처리를 위하여 비단말 노드 분할시 전체 영역을 최대로 감소시킬 수 있는 엔트리를 분할 대상으로 선정하여 비단말 노드의 사장 영역을 감소시킬 수 있는 삽입 및 분할 정책인 개선된 최대 영역 감소 정책을 제안하여 복합 질의에서 우수한 성능을 나타내는 TB-tree에 이를 적용하여 TB-tree와의 성능을 비교한다.

이 논문의 구성은 다음과 같다. 2장에서 관련 연구를 기술하고, 3장에서는 TB-tree의 특징과 공간 또는 시간 도메인 위주로 분할하였을 때 발생하는 문제점을 기술한다. 4장에서는 3장의 문제점을 해결할 수 있는 개선된 최대 영역 감소 정책을 제시하고, 5장에서는 개선된 최대 영역 감소 정책과 최근 시간 분할 정책의 영역 질의에 대한 성능평가를 보인다. 6장에서는 결론 및 향후 연구를 기술한다.

2. 관련연구

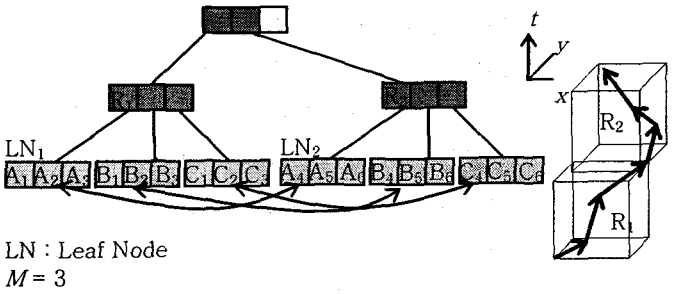
Pfoser[3]는 기존의 공간 색인이나 시간 색인에서 사용하는 영역 질의와 타임 슬라이스 질의와 같은 좌표 기반 질의(coordinate-based query) 이외에도 궤적을 위한 질의를 정의하였다. 궤적 기반 질의는 enter, leave, cross, bypass와 같은 위상 질의(topological query)와 움직인 거리, 속도, 방향과 같은 항해 질의(navigational query)로 분류하며, 궤적을 선택하는 과정과 궤적을 추출하는 두 단계로 이루어진 복합 질의도 정의하였다. Zhu[4]는 궤적 질의를 위상 술어(topological predicate)와, 속도와 방향과 같은 움직임 에 대한 추가적인 조건으로 구분하여 나타내었다.

이동체 색인으로 영역 질의에 효과적인 궤적 색인으로 SETI[5] 등이 있다. SETI는 공간적으로 잘 나뉘어진 셀(cell)과 각 셀에 시간 색인이 있는 2단계 구조로 되어 있어 하나의 셀 안에 포함되는 작은 영역 질의는 효과적이다. 그러나 한 선분이 여러 개의 셀에 해당되는 경우 셀 마다 중복 저장하므로 셀을 둘 이상 포함하는 영역 질의에는 질의 영역에 포함되는 셀을 모두 검색한 다음 중복을 제거하는 과정의 추가 비용이 필요한 문제점이 있다.

영역 질의와 복합 질의를 동시에 고려한 궤적 색인으로 STR-tree[3]와 OP-tree[4] 등이 있다. STR-tree는 R-tree 기반이며 공간 근접성뿐만 아니라 궤적 보존을 시도하는 색인 구조이지만 R-tree 보다 영역 질의의 비용이 높고, TB-tree 보다 복합 질의의 비용이 높다. OP-tree는 객체 단순화 방법으로 기존의 일반적으로 사용하는 MBB 대신 MBOP (Minimum Bounding Octagon Prism)을 사용한다. MBOP가 MBB보다 더 세밀하게 객체를 단순화 시킬 수 있으므로 MBOP는 MBB에 비하여 사장 영역을 더 적다. 그러나 MBOP는 MBB보다 더 많은 정보를 사용해야 하므로 노드의 팬아웃이 낮아지는 단점이 있다.

3. 문제 정의

이 장에서는 TB-tree의 특징과 TB-tree의 최근 시간 분할 정책으로 인하여 발생하는 문제점에 대하여 기술한다.



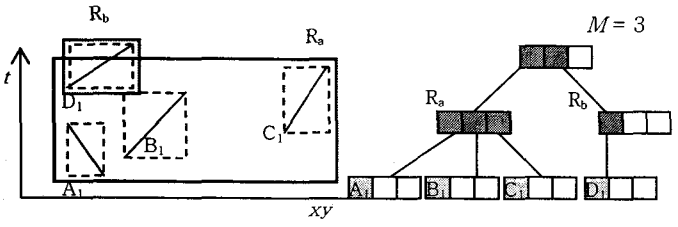
LN : Leaf Node
$M = 3$

그림 1. TB-tree 구조

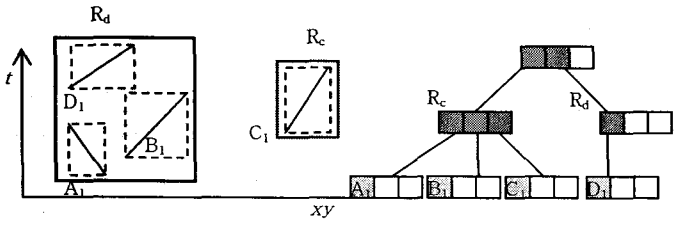
TB-tree는 이동체의 위치를 샘플링하여 선형 보간법(linear interpolation)을 사용하여 궤적을 표현한다. TB-tree는 그림 1과 같이 하나의 단말 노드에 동일한 이동체의 궤적만을 저장하는 궤적 보존 정책을 사용하며, 그림 1의 LN_1과 LN_2와 같이 동일한 이동체의 궤적을 저장하고 있는 단말 노드간의 연결 리스트가 존재하는 색인 구조를 가진다. 연결 리스트를 사용하는 논리적 궤적 표현 구조를 사용하여 궤적 추출을 효과적으로 할 수 있다.

TB-tree는 분할 정책으로 최근 시간 분할 정책을 사용한다. 최근 시간 분할 정책은 노드 N이 오버플로우 되었을 때, 가장 최근 시간의 엔트리만 새로운 노드에 저장한다. 따라서 그림 1의 LN_1이 가득 차 있을 때 A_4가 삽입되어 오버플로우가 발생하면, 시간적으로 가장 최근의 엔트리인 A_4만 새로운 노드(LN_2)에 저장된다. 따라서 TB-tree는 R-tree의 최소 영역 확장(Least Area Enlargement) 정책과 달리 시공간적으로 인접한 두 이동체의 궤적은 서로 다른 단말 노드에 저장될 뿐만 아니라 비단말 노드에서도 시간 순으로 저장되므로 궤적 간의 공간적 근접성을 고려하지 못한다. 그러나 공간 근접성을 고려하기 위하여 단말 노드를 분할하는 정책은 궤적 보존 정책에 위배되어 궤적 추출 비용을 증가시키게

된다.



(a) 시간 도메인만 고려한 분할 방식



(b) 공간 도메인을 고려한 분할 방식

그림 2. 비단말 노드의 겹침과 사장 영역

TB-tree의 최근 시간 분할 정책은 공간적인 특성보다 시간적 순서로 색인을 구성하므로 비단말 노드에서 겹침과 사장 영역이 커질 수 있다. 그림 2(a)는 비단말 노드의 팬아웃(fanout)이 3인 TB-tree에 시간대가 비슷한 이동체 A, B, C, D의 궤적 선분 A_1, B_1, C_1, D_1이 순서대로 삽입된 것을 표현한 것이다. 비단말 노드 R_a는 궤적 선분 A_1, B_1, C_1을 포함한 단말 노드들로 구성되고, 비단말 노드 R_b는 최근에 들어온 궤적 선분 D_1을 포함한 단말 노드를 구성하게 되어, R_b는 R_a에 겹침이 심하고 R_a는 사장 영역이 매우 크다.

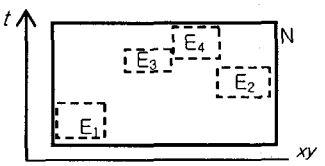
반면 그림 2(b)는 삽입 순서가 아닌 공간 근접성을 고려하여 생성된 경우이다. 궤적 선분 C_1이 가장 공간 근접성이 낮으므로 궤적 선분 A_1, B_1, D_1이 비단말 노드 R_c를 구성하게 되고, 궤적 선분 C_1이 비단말 노드 R_d를 구성하게 되어, 두 노드 R_c와 R_d의 겹침이 없고, R_c의 사장 영역도 그림 2(a)의 R_a에 비하여 작으므로 영역 질의에 효율적이다. 따라서 궤적 색인은 그림 2(b)와 같이 공간 근접성을 고려해야 한다.

4. 삽입 정책

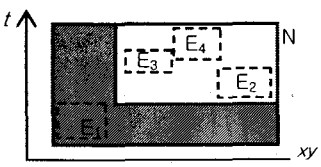
이 장에서는 TB-tree의 궤적 보존 및 공간활용도의 장점은 유지하면서 3장의 비단말 노드에서의 겹침과 사장 영역 문제를 개선할 수 있는 삽입/분할 정책과 알고리즘을 제시한다.

4.1 최대 영역 감소 정책

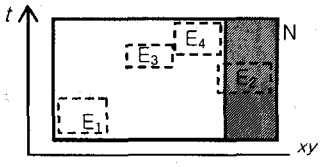
MAR(Maximum Area Reduction) 정책[6]은 TB-tree의 비단말 노드의 사장 영역과 겹침 문제를 해결하고자 한 비단말 노드 분할 정책이다. MAR 정책은 비단말 노드 N이 오버플로우 되었을 때, 분할 후의 N이 가장 작은 영역을 가지도록 분할한다.



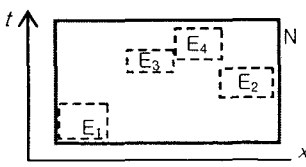
(a) 오버풀된 노드 N(M=3)



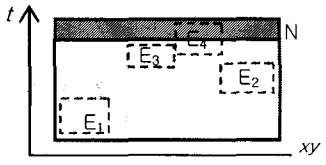
(b) E_1 제거후의 N



(c) E_2 제거후의 N



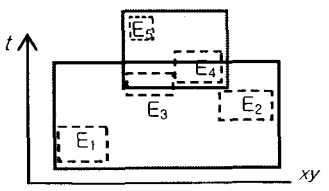
(d) E_3 제거후의 N



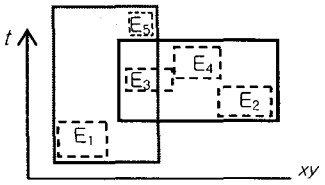
(e) E_4 제거후의 N

그림 3. MAR 정책의 분할 방법

MAR 정책의 비단말 노드 분할 방법은 그림 3(a)과 같이 비단말 노드 N이 오버플로우 되었을 때 N의 엔트리를 하나씩 제거한 다음 N의 면적(부피)을 계산하여, 제거 되었을 때 N의 면적을 가장 최소로 만드는 엔트리를 분할 대상으로 선정하여 새 노드에 저장한다. 따라서 그림 3의 경우 E_1이 제거 되었을 때 N의 영역이 가장 최소가 되므로 E_1이 분할 대상으로 선정되어 새로운 노드에 저장된다.



(a) 최근 시간 분할



(b) 최대 영역 감소 분할

그림 4. MAR 정책의 문제점

그러나 MAR 정책은 시간 도메인의 겹침과 사장 영역을 크게 만드는 문제가 있다. 그림 4는 그림 3(a)에서 최근 시간 분할 정책과 MAR 정책의 차이와 MAR의 문제점을 나타낸다. 그림 4(a)는 최근 시간 분할 정책에 따라

엔트리 E_4를 분할 대상으로 선정하여 분할한 뒤, 새로운 엔트리 E_5가 삽입되었을 때의 TB-tree를 나타낸 것이다. 반면 그림 4(b)는 최대 영역 감소 정책에 따라 그림 3(a)에서 전체 영역을 가장 최소로 만드는 엔트리 E_1이 분할 대상으로 선정된 후 새로운 엔트리 E_5의 삽입 후를 나타낸 것이다. 그림 4(b)가 그림 4(a)에 비하여 시간 도메인의 겹침이 심하고 사장 영역이 커 영역 질의의 성능을 저하시킨다.

시간 도메인 겹침 심화 문제는 그림 4(b)의 E_1과 같이 노드의 과거 영역이 분할 대상으로 선정됨으로써 발생한다. 따라서, 시간과 공간을 고려해서 영역을 최소화 하는 정책이 필요하다.

4.2 개선된 최대 영역 감소 정책

개선된 최대 영역 감소 정책(eMAR : enhanced Maximum Area Reduction)은 MAR 정책의 시간 도메인 겹침 심화 문제(그림 4)를 해결하기 위하여 MAR 정책에 시간적 제약 조건을 추가한 것이다. 시간적 제약 조건은 과거 영역의 엔트리가 분할 대상이 되지 않도록 하기 위하여, 유효 시간을 만족하는 엔트리만 분할 대상 후보가 될 수 있도록 한다.

유효 시간대(effective time span)는 일정 시간 범위로써, 임의의 엔트리의 시작 시간이 유효 시간 이내에 있다면 그 엔트리는 분할 대상의 후보가 된다. 유효 시간대를 설정하기 위하여 유효 시간대 파라미터 p를 사용한다. 관련된 용어의 정의는 아래와 같다.

N : 분할이 필요한 노드
E_i : N의 i번째 엔트리
p : 유효 시간대 파라미터 ($0 \le p \le 1$)
ET(N|E_i) : 노드 또는 엔트리의 최종시간(end time)
ST(N|E_i) : 노드 또는 엔트리의 시작시간(start time)

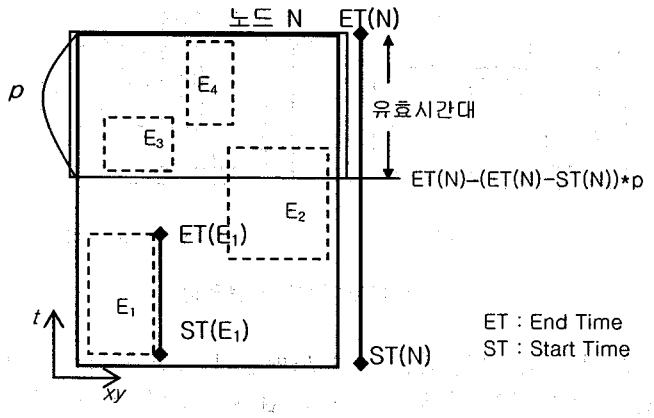


그림 5. 유효 시간대

노드 N의 유효시간은 노드의 최종 시간(ET(N))에서 ET(N) - (ET(N)-ST(N))*p 까지 이다. MAR 정책과 달리 eMAR은 ST(E_i) ≥ ET(N) - (ET(N) - ST(N))*p의 조건을 만족하는 엔트리만 후보로 하여 분할 대상을 선정한다. 그림 5에서는 E_3와 E_4만 유효시간 이내에 있으므로 분할 대상 후보가 된다.

만약 p가 1이라면, ST(E_i) ≥ ST(N)으로 되어 어떤 엔트리도 분할 대상으로 선정될 수 있으므로, MAR 정책과 동일하다. 반면 p가 0인 경우 ST(E_i) ≥ ET(N)이 되어 분할 대상을 선정할 수 없게 된다. 이 경우 TB-tree와 같이 최근 시간 분할 정책을 따른다. 따라서 p가 0인 경우는 TB-tree와 동일하다.

그러나 MAR 정책과 eMAR 정책은 분할 발생시 기존 TB-tree의 최근 시간 분할 정책에 비하여 연산량이 많으므로 삽입 속도가 저하되는 문제가 있다. 따라서 분할 대상 엔트리를 선정하기 위해 필요한 연산량을 줄여 삽입 속도를 개선할 필요가 있다. 이를 위하여 MBB 면에 접하지 않는 엔트리는 제거하여도 MBB의 면적(부피)은 줄어들지 않는다는 특성을 이용한다.

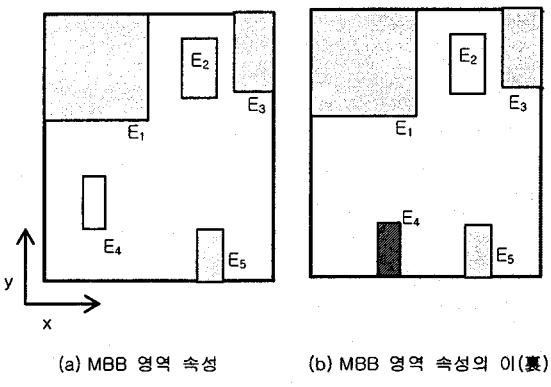


그림 6. MBB 영역 속성

그림 6(a)에서 E_1, E_3, E_5는 제거하면 면적이 줄어들지만, E_2, E_4는 MBB 면에 접하지 않으므로 제거하여도 MBB의 면적은 줄어들지 않는다. 그러나 MBB 면에 접한 엔트리를 제거하였다고 해서 반드시 MBB의 면적이 줄어드는 것은 아니다. 그림 6(b)의 E_4는 제거하더라도 E_5로 인하여 MBB의 면적은 줄어들지 않는다.

이와 같이 영역에 대한 특성을 이용하여 모든 엔트리를 대상으로 후보를 선정하는 것이 아니라 영역 감소가 발생하지 않는 엔트리를 제외한 나머지 엔트리만 분할 대상의 후보로 설정하여, 후보의 수를 줄여 삽입 성능을 개선할 수 있다.

4.3 알고리즘

개선된 최대 영역 감소 정책과 최근 시간 분할 정책의 차이점은 비단말 노드 분할시 분할 대상 선정 방법에 있

다. TB-tree의 최근 시간 분할 정책은 가장 최근에 삽입된 엔트리를 분할대상으로 설정하는 반면, 개선된 최대 영역 감소 정책의 경우는 유효 시간대 내에서 임의의 엔트리를 제거하였을 때 노드의 영역을 가장 최소로 만드는 엔트리를 분할 대상으로 선정한다.

```
Algorithm Insert(N, E)
INS1  Invoke FindNode(N, E) to select a leaf node N' which has a predecessor of E
INS2  If node N' is found,
        Insert E
      else
        Create a new leaf node N" for E
        Insert E
INS3  AdjustTree(N')
```

알고리즘 Insert는 색인에 새로운 엔트리 E를 삽입하는 알고리즘이다. 먼저 FindNode를 수행하여 이전에 보고된 위치가 있는지 확인한 다음, 보고된 위치가 있다면 그 노드에 삽입을 수행하여 궤적을 보존한다. 보고된 위치가 없다면 새로운 이동체가 처음으로 보고된 것이므로 새로운 이동체를 위하여 새로운 단말 노드를 생성한다. 삽입한 뒤에는 AdjustTree를 호출하여 변경된 MBB를 상위 노드로 전파해야 하며 필요한 경우 오버플로우를 처리하도록 한다.

```
Algorithm AdjustTree(N)
AT1  If N is the root
       Return
AT2  If N is overflow
       Invoke Split(N)
AT3  Let P be the parent node of N, and let E_N be N's entry in P
AT4  Adjust MBB(E_N) so that it tightly encloses all entry boxes in N
AT5  Set N = P and repeat from AT1
```

알고리즘 AdjustTree는 삽입이나 분할로 인하여 변경된 MBB를 상위 노드에 전파하는 알고리즘이며 오버플로우가 발생하여 분할을 해야 할 경우 Split을 호출하여 분할을 처리하도록 한다.

```
Algorithm Split(N)
S1   If N is a leaf node
       Create new leaf node
       Make trajectory link between old node and new one.
     else
       Invoke SplitNonleafNode(N)
```

알고리즘 Split은 분할이 발생한 노드 N이 단말 노드 또는 비단말 노드인지 확인하여 단말 노드인 경우 기존의 TB-tree와 마찬가지로 최근 시간 분할 정책을 사용하여 분할하고 분할 전후의 두 노드에 대하여 궤적 연결 리스트를 생성한다. 비단말 노드인 경우 단말 노드와 다른 분할 정책을 사용하므로 SplitNonleafNode를 호출하여 처리한다.

```
Algorithm SplitNonleafNode(N)
SNN1  For each E which is satisfied MBB area property and effective time span, calculate d = volume(N) - volume(N-E)
SNN2  Select E' which makes largest d
SNN3  If rightmost node N" whose level is same as N has space
        Insert E' to N"
      else
        Create a new non-leaf node for E'
SNN4  AdjustTree(N")
```

알고리즘 SplitNonleafNode는 최근 시간분할 정책과 최대 영역 감소 정책 및 개선된 최대 영역 감소 정책의 실질적인 차이가 존재하는 알고리즘이며 비단말 노드 N이 오버플로우 되었을 때 호출된다. MBB 영역 속성을 만족하는 엔트리중 유효 시간 이내에 있는 엔트리를 후보로 설정한 뒤, 각 후보 엔트리 E에 대하여 노드 N의 면적과 후보 엔트리 E를 제거한 노드 N' 의 면적 차 d를 계산한다. 영역 감소가 가장 큰 d를 가지는 엔트리를 분할 대상으로 선정한다. 선정된 분할 대상에 대해서는 TB-tree와 동일하게 색인의 가장 오른쪽 패스(rightmost path)에 저장한다.

5. 성능 평가

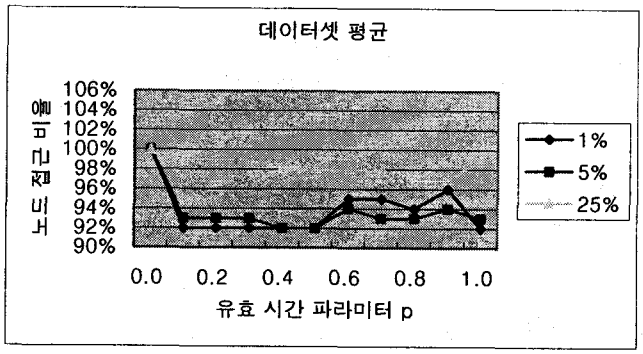
이 장에서는 개선된 최대 영역 감소 정책을 TB-tree에 적용하여 영역 질의시의 노드 접근 횟수를 성능으로 평가한다.

성능 평가는 노드에 접근할 때마다 노드 I/O를 계산하였으며, 캐쉬(cache)는 없는 것으로 가정하였다. 데이터셋은 GSTD[7]를 기반으로 하여 생성하였다. GSTD로 생성한 데이터셋은 시공간의 점으로 이동체의 궤적 정보를 나타내지만, 성능 평가에 적용하기 위하여 이동체의 궤적은 시공간의 선분으로서 색인에 삽입되어야 하므로 GSTD에 의해 생성된 데이터셋에서 동일 궤적 ID를 가진 시간 순서화 된 두 점을 하나의 선분으로 사용한다. 사용한 데이터셋은 초기 가우시안 분포에서 시간이 흐름에 따라 다양한 방향으로 이동체가 이동하는 분포를 사용하였다. 각각의 데이터셋에 대하여 이동체의 수는 100개로, 각 이동체당 보고 횟수도 1000회로 고정하였다.

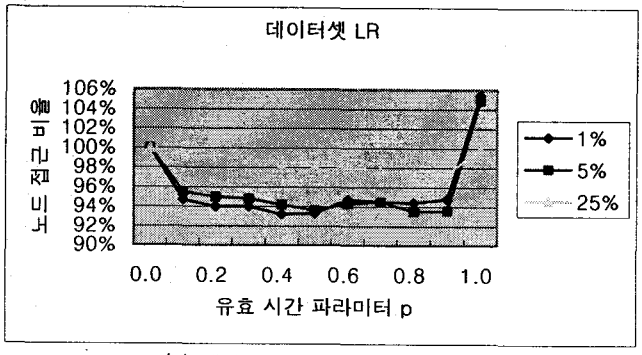
위의 데이터셋들에 대하여 시공간의 각 축에 대하여 1%, 5%, 25% 의 시공간 영역 질의를 각각 1000회 반복하여 노드 접근 횟수를 측정한 뒤 평균하였다.

TB-tree의 최근 시간 분할 정책(p=0)의 영역 질의시 노드 접근 횟수를 100%로 두어 개선된 최대 영역 감소 정책의 유효 시간 파라미터 $p(0 \le p \le 1)$를 달리하여 성능을 비교한다. 비율이 낮을수록 TB-tree에 비하여 더

적은 노드 접근을 한 것을 의미한다.



(a) 평균 데이터셋에 대한 영역 질의 비교



(b) 데이터셋 LR에 대한 영역 질의 비교

그림 7. 영역 질의 결과

그림 7(a)는 각각의 데이터셋에 대하여 유효 시간 파라미터 p의 변화에 따른 영역 질의 결과의 평균이다. 전체적으로 개선된 최대 영역 감소 정책(eMAR)이 최근 시간 분할 정책(p=0) 또는 최대 영역 감소 정책(p=1)에 비하여 영역 질의에서 TB-tree 대비 약 6~8%의 질의 성능 향상 효과가 있다.

축당 1% 시공간 영역 질의와 같이 질의 크기가 작은 경우에는 MAR 정책과 eMAR 정책이 유사하게 TB-tree 대비 최대 약 9%의 성능 향상을 나타낸다. 축당 5%의 시공간 영역 질의의 경우 약 p=0.4인 경우 가장 우수한 성능을 나타내어 MAR 정책보다 다소 우수한 성능을 나타낸다. 그러나 축당 25%의 시공간 영역 질의의 경우 p=0.9일 때 가장 우수한 성능을 보이며, MAR 정책은 최근 시간 분할 정책에 비하여 성능 향상 효과가 적다. 이는 질의의 크기가 큰 경우 각 비단말 노드의 사장 영역이나 겹침이 줄어드는 것에 대해 더 적은 영향을 받기 때문에 MAR 정책의 효과가 적은 것으로 판단된다.

그림 7(b)의 경우는 이동체가 왼쪽에서 오른쪽으로 이동하는 분포이다. 각각의 영역 질의에 대하여 MAR 정책이 가장 좋지 않은 성능을 내는 경우로, 과거 영역의 엔트리가 분할 대상으로 선정되어 시간 도메인의 겹침 심화 현상이 나타난 것으로 판단된다.

6. 결론 및 향후 연구

이 논문에서는 기존의 복합 질의에 효율적인 TB-tree의 비단말 노드에서의 겹침과 사장 영역이 커지는 문제를 정의하고, 이 문제를 해결하기 위하여 사장 영역을 줄일 수 있는 비단말 노드 분할 정책인 개선된 최대 영역 감소 정책을 제안했다.

개선된 최대 영역 감소 정책은 궤적을 번들하는 TB-tree의 특성을 유지하므로 복합 질의에 우수한 성능을 나타내며, 공간 활용도가 높아 대용량의 이동체 궤적 데이터 색인으로 적합하다. TB-tree에 비하여 비단말 노드의 사장 영역 및 겹침을 줄여 영역 질의의 비용이 감소되었음을 알 수 있다. 시공간 축당 1%, 5%와 같은 작은 영역 질의에 대해서는 p가 약 0.4 일 때 가장 우수한 성능을 보이며, 시공간 축당 25%와 같은 큰 영역 질의에 대해서는 p를 0.9로 설정하는 것이 영역 질의에 효과적인 것을 실험을 통하여 확인하였다.

향후 연구로써 좀 더 다양한 데이터셋에 대한 실험과 서로 다른 이동체 분포에 적합한 유효 시간 파라미터에 대한 연구가 필요하다.

7. 참고문헌

[1] Yannis Theodoridis, Michael Vazirgiannis, Timos Sellis, "Spatio-Temporal Indexing for Large Multimedia Applications", IEEE International Conference on Multimedia Computing and Systems, pp 441-448, 1996.

[2] Antonm Guttman, "R-Trees: A Dynamic Index Structure for Spatial Searching", Proceedings of ACM-SIGMOD Conference on the Management of Data, pp. 47-57, 1984

[3] Dieter Pfoser, Christian S. Jensen, Yannis Theodoridis, "Novel Approaches to the Indexing of Moving Objects", Proceedings of International Conference on Very Large Data Bases, pp. 395-406, 2000

[4] Hongjun Zhu, Jianwen Su, Oscar H. Ibarra, "Trajectory Queries and Octagons in Moving Object Databases", Proceedings of the ACM Conference on Information and Knowledge Management, pp. 413-421, 2002

[5] V. Prasad Chakka, Adam C. Everspaugh, Jignesh M. Patel, "Indexing Large Trajectory Data Sets With SETI", Proceedings of the 2003 CIDR Conference, 2003

[6] 장종우, 임덕성, 홍봉희, "시공간 근접성을 고려한 TB-tree의 동적 삽입 정책", 정보과학회 2003 춘계 학술대회 논문집(30권 1호), pp 776-778, 2003

[7] Yannis Theodoridis, Jefferson R. O. Silva, Mario A. Nascimento, "On the Generation of Spatiotemporal Datasets", Advanced in Spatial Databases, pp 147-164, 1999