이동 객체 경로 탐색을 위한 시공간 클러스터링 기법†

A Spatio-Temporal Clustering Technique for the Moving Object Path Search

이기영*, 강홍구**, 윤재관**, 한기준**

Ki-Young Lee, Hong-Koo Kang, Jae-Kwan Yun, Ki-Joon Han

요 약 최근 들어 지리 정보 시스템이 발전함에 따라 경로 검색, 주변 정보 검색, 응급 서비스 등을 제공하는 위치 기반 서비스, 텔레매틱스 등의 새로운 응용 서비스 개발에 대한 관심과 연구가 증대되고 있다. 위치 기반 서비스 및 텔레매틱스에서 사용되는 시공간 데이타베이스에서의 사용자의 검색은 시간 축을 현재의 시간으로 고정하고 공간 및 비공간 속성을 검색하기 때문에 시간 축에 대한 검색 범위가 넓을 경우에는 이를 효율적으로 처리하기 어렵다. 이를 해결하기 위하여 이동 객체의 위치 데이타를 요약하는 기법인 스냅샷이 소개되었다. 그러나, 이러한 스냅샷 기법은 저장해야 되는 공간 영역이 넓을 경우 저장 공간이 많이 필요하며 검색에 자주 사용되지 않는 불필요한 영역까지 스냅샷을 생성하므로 저장 공간 및 메모리를 많이 사용하게 된다.

이에 본 논문에서는 기존의 스냅샷 기법의 단점을 극복하기 위하여 이전에 공간 클러스터링을 위해 사용되던 2차원의 공간 해시 알고리즘을 시공간으로 확장한 해시-기반 시공간 클러스터링 알고리즘(H-STCA)과 과거 위치 데이타로부터 이동 객체 경로 탐색을 위한 지식을 추출하기 위해 H-STCA 알고리즘에 근거한 지식 추출 알고리즘을 제안한다. 그리고, 대용량의 이동 객체 데이타에 대한 검색 시간, 저장 구조 생성 시간, 최적 경로 탐색 시간 등에서 H-STCA를 사용한 스냅샷 클러스터링 방법, 기존의 시공간 인덱스 방법, 스냅샷 방법과의 성능평가에 대하여 설명한다. 성능평가 결과로 H-STCA를 사용한 스냅샷 클러스터링 방법은 기존의 시공간 인덱스 방법이나 스냅샷 방법 보다 이동 객체의 개수가 증가하면 할수록 성능 향상이 더욱 큰 것으로 나타났다.

Abstract Recently, the interest and research on the development of new application services such as the Location Based Service and Telemetics providing the emergency service, neighbor information search, and route search according to the development of the Geographic Information System have been increasing. User's search in the spatio-temporal database which is used in the field of Location Based Service or Telemetics usually fixes the current time on the time axis and queries the spatial and aspatial attributes. Thus, if the range of query on the time axis is extensive, it is difficult to efficiently deal with the search operation. For solving this problem, the snapshot, a method to summarize the location data of moving objects, was introduced. However, if the range to store data is wide, more space for storing data is required. And, the snapshot is created even for unnecessary space that is not frequently used for search. Thus, more storage space and memory are generally used in the snapshot method.

Therefore, in this paper, we suggests the Hash-based Spatio-Temporal Clustering Algorithm(H-STCA) that extends the two-dimensional spatial hash algorithm used for the spatial clustering in the past to the three-dimensional spatial hash algorithm for overcoming the disadvantages of the snapshot method. And, this paper also suggests the knowledge extraction algorithm to extract the knowledge for the path search of moving objects from the past location data based on the suggested H-STCA algorithm. Moreover, as the results of the performance evaluation, the snapshot clustering method using H-STCA, in the search time, storage structure construction time, optimal path search time, related to the huge amount of moving object data demonstrated the higher performance than the spatio-temporal index methods and the original snapshot method. Especially, for the snapshot clustering method using H-STCA, the more the number of moving objects was increased, the more the performance was improved, as compared to the existing spatio-temporal index methods and the original snapshot method.

주요어 : LBS, 시공간 인덱스, 해시-기반 시공간 클러스터링, 이동 객체 경로 탐색

KeyWords : LBS, Spatio-Temporal Index, H-STCA, Moving Object Path Search

† 본 연구는 정보통신부 및 정보통신연구진흥원의 대학 IT 연구센터 육성·지원사업의 연구결과로 수행되었음.

* 서울보건대학 인터넷정보과 kylee@shjc.ac.kr

** 건국대학교 컴퓨터공학부 {hkkang, jkyun, kjhan}@db.konkuk.ac.kr

1. 서 론

최근 들어 휴대폰, PDA, 자동차 등과 같이 시간의 흐름에 따라 공간 및 비공간 속성이 빈번하게 변화하는 방대한 시공간 데이타를 관리하는 시공간 데이타베이스 기술과 GPS(Global Positioning System)를 탑재한 이동 단말기를 이용한 이동 객체의 위치 추적 장치와 CDMA, WiFi, WiBro 등과 같은 무선 통신 기술이 급속히 발전하고 있다. 또한, 지리 정보 시스템(GIS: Geographic Information System)이 발전함에 따라 경로 검색, 주변 정보 검색, 응급 서비스 등을 제공하는 위치 기반 서비스(LBS:Location Based Service), 텔레매틱스(Telemetics) 등의 새로운 응용 서비스 개발에 대한 관심과 연구가 증대되고 있다[1],[2],[3].

이동 객체의 이동과 관련된 위치 기반 질의를 처리하기 위해서는 이동 객체의 위치 데이타를 이동 객체 데이타베이스 관리 시스템에 저장하고 효율적인 질의 처리를 위해 이동 객체 인덱스가 필요하다. 일반적으로 시공간 지식 탐색에서 사용되는 데이타는 시간 축에 따른 공간 및 비공간 데이타를 사용하기 때문에 검색되는 시간 축의 양이 많을수록 데이타의 양이 기하급수적으로 증가한다. 그러므로, 검색되는 시간이 좁은 범위일 경우에는 위치 인덱스를 사용하는 것이 효과적일 수 있지만 검색되는 시간의 범위가 넓으면 위치 인덱스를 사용하기 어렵다.

이를 효율적으로 해결하기 위하여 이동 객체의 위치 데이타를 요약하는 기법에는 스냅샷(Snapshot) 기법이 있다. 이러한 스냅샷 기법의 활용 방법에는 다음 두 가지가 있다[4],[5]. 첫째, 위치 데이타를 획득하는 시점부터 스냅샷을 생성하여 두고 시공간 질의를 처리하는 방법이다. 이 방법은 검색하고자 하는 시간 축에 대한 공간 데이타가 이미 생성되어 있으므로 시공간 질의 처리를 빠르게 할 수는 있으나, 저장해야 되는 공간 영역이 넓을 경우 저장 공간이 많이 필요하며 검색에 자주 사용되지 않는 불필요한 영역까지 스냅샷을 생성하므로 저장 공간 및 메모리를 많이 사용하게 된다. 둘째, 시공간 질의를 처리하는 시점에 필요한 스냅샷만을 생성하여 시공간 질의를 처리하는 방법이 있다. 이 방법은 시공간 질의 처리가 쉽고 저장 공간도 절약되며 단기간의 지식 추출에 효과적이지만, 장시간의 시공간 질의일 경우에는 필요한 스냅샷들을 생성하는 비용이 크게 된다.

본 논문에서는 이러한 스냅샷 기법의 단점을 극복하기 위하여 이전에 공간 클러스터링을 위해 사용되던 2차원의 공간 해시 알고리즘을 시공간으로 확장한 해시-기반 시공간 클러스터링 알고리즘(H-STCA)과 제안한 H-STCA 알고리즘에 근거한 과거 위치 데이타로부터 이동 객체 경로 탐색을 위한 지식을 추출하기 위해 지식 추출 알고리즘을 제안하였다. 또한, 본 논문에서 제안한 시공간 클러스터링 알고리즘의 효율성을 입증하기 위해 성능 평가도 수행하였다.

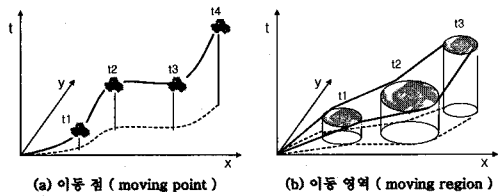
본 논문의 구성은 다음과 같다. 제 2 장에서는 관련 연구로서 이동 객체 모델링, 위치 데이타 인덱스, 그리고 이동 객체의 과거 위치 요약에 관련한 스냅샷에 대하여 살펴본다. 제 3 장에서는 본 논문에서 제시한 H-STCA 알고리즘, 시공간 클러스터링 알고리즘과 스냅샷을 사용하여 이동 객체 경로 탐색을 위한 지식 추출 알고리즘을 제시한다. 제 4 장에서는 본 논문에서 제안한 시공간 클러스터링 알고리즘의 최적 임계치를 구하기 위한 비교 실험에 대하여 언급한다. 또한, 기존 과거 위치 인덱스를 이용하여 질의를 처리하는 성능과 스냅샷만을 이용하여 질의를 처리하는 성능과 비교한 성능 평가에 대하여 기술한다. 마지막으로 제 5 장에서는 결론에 대해 언급한다.

2. 관련 연구

본 장에서는 이동 객체 모델링에 대해 살펴보고, 이동 객체의 과거 위치 질의에 대한 기존 인덱스 방법 및 이동 객체의 과거 위치 데이타 요약에 관련한 스냅샷에 대하여 설명한다.

2.1 이동 객체의 모델링

이동 객체는 시간이 흐름에 따라 객체의 위치나 영역과 같은 공간 정보가 연속적으로 변화하는 객체를 말하며 크게 이동 점(moving point)과 이동 영역(moving region)으로 나눌 수 있다[6].



(a) 이동 점 (moving point)　(b) 이동 영역 (moving region)

<그림 1> 이동 점과 이동 영역

이동 점은 <그림 1>(a)에서 보여주는 바와 같이 시간에 따라 객체의 위치가 변하는 것으로 이러한 이동 점의 예는 모바일 사용자, 택시, 배, 비행기 등이 있다. 이동 점에 대한 질의는 이동 객체의 위치에 대한 질의이며 이동 객체가 가지는 면적에 대해서는 질의를 수행하지 않는다. 예를 들어 "2004년 8월 20일 오후 1시 30분에 서울2커 8468 차량은 어느 위치에 있는가?"와 같이 이동 점의 위치를 묻는 질의의 결과 값은 그 시점의 이동 점 객체의 위치를 반환한다.

이동 영역은 <그림 1>(b)에서와 같이 시간에 따라 객체의 위치뿐만 아니라 모양까지 변하는 것으로 한 국가의 행정 구역이나 오존층의 변화, 폭풍의 이동 경로, 암세포의 상태 등을 예로 들 수 있다. 이러한 이동 영역 객체는 시간에 따라 위치와 모양이 변하며 특정 시점에 대한 질의는 그 시점에 이동 영역이 존재하는 위치와 모양을 표현하는 영역을 반환한다. 예를 들어, "태풍이 가장 활발한 움직임을 보일 때 태풍의 크기를 보이시오."와 같이 이동 영역에 대한 질의는 특정 시간에 이동 객체의 위치와 모양에 대한 질의이다.

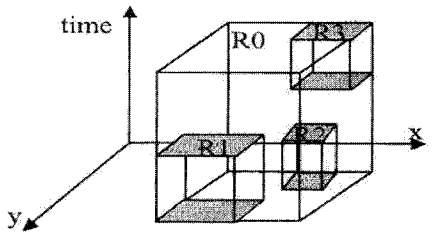
시간에 따라 위치가 연속적으로 변하는 이동 객체를 표현하기 위한 데이타 모델은 연속적 모델과 이산적 모델로 구분된다. 연속적 모델은 시공간 이동 객체를 무한한 점들의 집합으로 표현하고, 이산적 모델은 이동 객체를 유일한 점들의 집합으로 표현한다. 기존의 데이타베이스 관리 시스템으로 이동 객체를 직접 표현할 경우 이동 객체의 위치가 계속적으로 변하므로 이를 기록하기 위한 빈번한 연산이 요구되고, 이동 객체의 현재 상태만을 기록하므로 과거 및 향후 이동 위치와 관련된 질의에 적절히 응답할 수 없다.

2.2 위치 데이타 인덱스

본 절에서는 과거 위치에 대한 인덱스 방법인 3D R-tree, MVR-tree, MV3R-tree에 대하여 분석한다.

2.2.1 3DR-tree

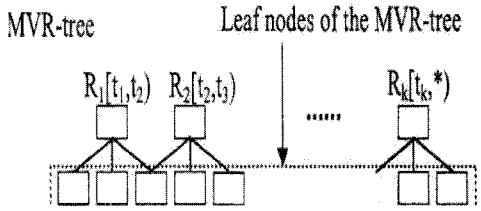
3DR-tree(Three Dimensional R-tree)[7]는 시간을 이차원 공간 내의 또 다른 차원으로 간주하여 이차원 영역(region)들과 시간 축을 고려한 3차원 MBR 형태로 표현한다. 또한, 이동 객체의 위치나 모양이 변경될 때마다 이 시간 동안 변경된 정보를 나타내는 MBR을 3DR-tree에 삽입한다. 시간이 경과함에 따라 3DR-tree가 다루는 시간 영역의 범위가 증가하기 때문에 Time-Slice 질의에 대한 성능은 크게 떨어질 수밖에 없으며, 장기간의 넓은 영역에 대한 정보를 처리하는 경우 비효율적이다. 게다가 위치나 모양 변경이 거의 없는 이동 객체들은 3DR-tree에 커다란 빈 공간(dead space)을 야기하며, 이러한 빈 공간은 3DR-tree의 성능을 크게 떨어뜨린다. 그럼에도 불구하고 3DR-tree는 불필요한 정보가 없기 때문에 인덱스 크기가 가장 작고, Time-Interval 질의에 대해서는 좋은 성능을 보인다는 장점이 있다. <그림 2>는 3DR-tree의 일반적인 구조를 나타낸 그림이다.



<그림 2> 3DR-tree의 구조

2.2.2 MVR-tree

MVR-tree(Multi-Version R-tree)[8]는 2가지 분할 방법을 사용한다. 하나는 버전 분할이고, 다른 하나는 키 분할이다. 버전 분할은 시간 축을 기준으로 노드를 자르는 방법이고, 키 분할은 시간 축을 고려하지 않고 노드를 자르는 방법이다. <그림 3>은 MVR-tree의 일반적인 구조를 보여준다.

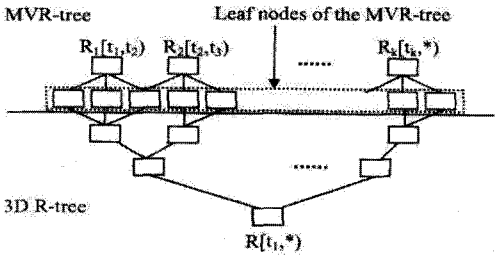


<그림 3> MVR-tree의 구조

MVR-tree는 버전 분할에서 발생하는 중복되는 데이타 때문에 인덱스 크기가 크다. 그러나, Time-Slice 질의 시에는 성능이 3DR-tree보다 우수하며 Time-Interval 질의 시에는 성능이 떨어진다. MVR-tree에서 엔트리의 구조는 <공간 MBR, 시작시간, 끝시간, 이동 객체포인터>이다.

2.2.3 MV3R-tree

MV3R-tree[9]는 MVR-tree와 3DR-tree를 결합한 구조이다. MVR-tree는 Time-Slice 질의를 처리할 때 사용되고, 3DR-tree는 Time-Interval 질의를 처리할 때 사용된다. <그림 4>는 MV3R-tree의 일반적인 구조를 나타낸 그림이다.



<그림 4> MV3R-tree의 구조

MV3R-tree는 몇 가지 단점을 가지고 있다. 버전 분할에서 발생하는 중복되는 데이타 때문에, MV3R-tree의 크기는 3DR-tree보다 대략 1.5배 정도 크다. 또한, 데이타의 삽입과 논리적 삭제 시에 MVR-tree와 3DR-tree를 모두 수정해야 하기 때문에, 변경 비용이 3DR-tree보다 대략 2배가 크다. Time-Slice 질의를 처리할 경우에도, MVR-tree가 관할하는 시간 간격이 크기 때문에 탐색 영역 또한 크게 된다. 그러므로, Time-Slice 질의에 대해서 MVR-tree는 더 많은 노드들을 접근해야 한다.

2.3 스냅샷 모델

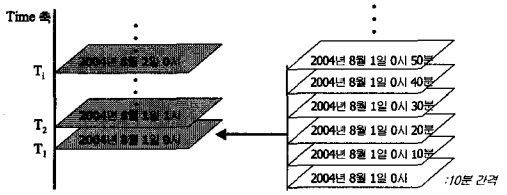
본 절에서는 이동 객체의 과거 위치 데이타에 대한 시공간 질의 시에 효율적인 시공간 질의를 처리하기 위하여 이동 객체의 위치 데이타를 요약하여 처리하는 스냅샷의 개념과 스냅샷 관련 질의에 대하여 설명한다.

2.3.1 스냅샷의 개념

시공간 데이타베이스는 현실 세계에 존재하는 다양한 시공간 객체들을 효과적으로 관리할 뿐만 아니라 시간의 흐름에 따라 변화하는 시공간 객체의 이력 또한 효율적으로 관리하기 위해 시공간 데이타 모델을 사용한다. 시공간 데이타 모델은 실세계에서 발생하는 데이타에 대한 시간 적용과 범주에 따라 스냅샷 데이타 모델(Snapshot Data Model), 시공간 복합 데이타 모델(Spatio-Temporal Composite Data Model) 등이 있다. 스냅샷 데이타 모델은 아주 간단한 시공간 데이타 모델 중의 하나로 어느 특정 시간에서 획득되는 시공간 레이어인 스냅샷을 표현한다. 시공간 복합 데이타 모델은 다수의 스냅샷 레이어들을 하나의 시공간 복합 레이어에 Overlapping하는 방법으로 스냅샷 데이타 모델을 확장한 것이다.

스냅샷은 이동 객체의 과거 위치 데이타에 대한 Time-Interval 동안에 임의의 특정 Time-Point에서 획득되는 레이어(Layer)로 정의된다[10]. 그러므로, 이동 객체의 과거 위치 데이타에 대한 스냅샷이란 특정 Time-Interval에서 그 시간 안에 존재하는 이동 객체의 과거 위치 데이타를 요약하여 하나의 레이어로 만든 것이다.

예를 들어, 2004년 8월 1일 0시에 특정 도로 구간에서 이동 객체 차량의 이동 대수를 구한다고 하자. 이동 객체의 위치 데이타를 획득하는 시간 단위가 10분 단위라 가정하고 스냅샷의 시간 단위는 1시간이라 한다면, 2004년 8월 1일의 0시부터 0시 50분까지의 이동 객체의 과거 위치 데이타를 요약하여야 한다. 이러한 스냅샷의 개념은 1980년대 후반에 지리 정보 시스템의 시공간 데이타 모델로부터 나왔고 최근에는 이동 객체의 과거 위치 데이타를 효율적으로 처리하기 위한 요약 정보를 갖는 스냅샷으로 발전하였다[4],[5].



<그림 5> 스냅샷의 구조

<그림 5>는 시간 축에 따라 임의의 특정 시간 T_1(2004년 8월 1일 0시), T_2(2004년 8월 1일 1시), . . . , Ti(2004년 8월 2일 0시)에서의 스냅샷에 대한 구조를 보여준다. 즉, T_1에서는 10분 간격으로 이동 객체의 위치 데이타를 획득하고 2004년 8월 1일 0시부터 0시 50분까지의 이동 객체의 과거 위치 데이타를 요약하여 하나의 레이어로 생성한 스냅샷을 보여주고 있으며, T_2에서도 10분 간격으로 이동 객체의 위치 데이타를 획득하고 2004년 8월 1일 1시부터 1시 50분까지의 이동 객체의 과거 위치 데이타를 요약하여 하나의 레이어를 생성한 스냅샷을 보여주고 있다. 마찬가지로

Ti에서도 동일한 방법으로 스냅샷을 생성한다.

2.3.2 스냅샷 관련 질의

스냅샷 관련 질의는 이동 객체 데이타베이스의 질의와 밀접한 관계가 있다. 이러한 이동 객체 데이타베이스의 질의 종류는 크게 영역 질의, 타임스탬프(Timestamp) 질의, 궤적 질의, 복합 질의로 나누어진다. 영역 질의는 주어진 시간 간격 동안에 공간 윈도우에 속하는 이동 객체들을 검색하는 3차원 질의이며, 타임스탬프 질의는 특정 시간에 주어진 공간 윈도우에 속하는 이동 객체들을 검색하는 질의이다. 궤적 질의는 특정한 이동 객체의 궤적을 검색하는 질의이며 복합 질의는 "특정 시간에 주어진 영역을 지나는 객체의 궤적을 검색하라"와 같이 시공간 도메인의 영역 질의와 궤적 질의가 복합된 질의이다[10].

이동 객체에 대한 질의에 대한 연구는 크게 CHOROCHRONOS[11]와 DOMINO[12] 및 국내 ETRI의 "대용량 위치 정보 관리 컴포넌트 기술 개발" 프로젝트[13]에서 볼 수 있다. CHOROCHRONOS의 접근 방법은 이동 객체의 과거에서부터 현재까지 움직인 궤적을 표현하는 데이타 형과 연산자를 정의하고 궤적의 위상 모델을 제시하였고, 이와 같은 모델을 이용하여 SQL을 확장한 이동 객체 질의 언어 STQL (Spatio-Temporal Query Language)를 제안하였다.

DOMINO는 기존의 공간 데이타베이스 관리 시스템을 이용하여 이동 객체의 현재 위치를 처리하고 이를 이용하여 미래의 위치를 예측하였다. 이러한 시스템은 이동 객체가 움직인 궤적보다 계속 현재의 위치 정보를 갱신해야 하므로 갱신 비용 절감에 관심을 두고 있다. 또한, 미래에 대한 예측에 관련된 FTL(Future Temporal Logic) 질의 언어를 제안하여 사용자의 질의를 표현하고 이를 처리할 수 있는 방법을 제시하고 있다.

ETRI의 "대용량 위치 정보 관리 컴포넌트 기술 개발" 프로젝트에서는 이동 객체 데이타베이스 관리 시스템의 사용자에게 인터페이스를 제공하고, 시간에 따라 변하는 위치에 대한 영역 질의, 최근접 질의, 네비게이션 질의 등을 처리하며, 클라이언트의 요청과 질의 컴포넌트의 인터페이스로써 SQL과 같은 형태의 질의어인 MOQL(Moving Object Query Language)을 제공하였다[13]. 이러한 MOQL에서는 이동 점 객체 모델 함수를 크게 Snapshot(), Slice(), Project() 함수 등으로 구성하였다. Snapshot() 함수는 이동 점의 특정 시점에서의 위치 정보를 반환하는 함수이고, Slice() 함수는 특정 시간 간격을 입력받아 해당 이동 점 객체를 반환하는 함수이다. Snapshot()과 Slice()의 가장 큰 차이점은 Snapshot() 함수는 이동 점 객체를 특정 시점의 현재 값으로 타입 변환을 하지만, Slice() 함수는 이동 점 객체의 타입 변환 없이 객체의 슬라이스를 반환한다는 것이다. Project() 함수는 위치 정보의 변화분에 대한 프로젝션 결과를 반환하는 함수이다.

3. H-STCA

본 장에서는 H-STCA(A Hash-based Spatio-Temporal Clustering Algorithm)의 데이타 구조, 알고리즘, 그리고 시공간 클러스터링 구조를 사용한 시공간 접근 방법에 대하여 설명한다. 마지막으로, H-STCA을 사용하여 클러스터링 및 스냅샷과 같은 지식을 발견하기 위한 지식 추출 알고리즘에 대해서도 기술한다

3.1 H-STCA 데이타 구조

3.1.1 시공간 해시 테이블 생성

H-STCA에서는 전체 시공간 도메인을 해시 함수를 통해 작은 범위로 축소시키게 된다. 예를 들면, 서울에서 부산으로 갈 수 있는 도로 지도를 가로, 세로의 일정한 영역으로 나누었을 때 각각의 격자는 시공간 해시 테이블의 하나의 셀(Cell)에 매핑될 수 있다. 즉, d-차원을 구성하는 각 i축(단, $1 \le i \le d$)을 m_i구간으로 나누어 전체 시공간 도메인을 $k = m_1 \times m_2 \times \ldots \times m_d$개 셀들의 집합으로 구성되는 시공간으로 변환한다. d-차원 공간의 임의의 좌표는 $(v_1, v_2, \ldots, v_d)$와 같이 표현된다. 시공간 해시 테이블에서는 각 축의 값에 대한 시공간 해시 함수 값들인 $(f_1(v_1), f_2(v_2), \ldots, f_d(v_d))$을 해당 좌표의 셀 주소(Cell Address)로서 사용한다. H-STCA에서 사용하는 i축을 위한 시공간 해시 함수 $f_i(v_i)$는 다음과 같다 (단, $1 \le i \le d$).

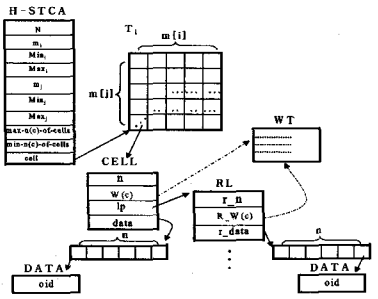
$$f_i(v_i) = \frac{v_i - \min_i}{\left\lceil \frac{\max_i - \min_i + 1}{m_i} \right\rceil}$$

$\max_i$와 $\min_i$는 각각 시공간 데이타베이스에 존재

하는 모든 공간 객체들의 i축의 값에 대한 최대값과 최소값이다. m_i는 i축을 구성하는 구간의 수이다. 즉, i축은 m_i개로 나뉜다. 그러므로, 셀 주소는 (0,0,...,0)~(m_1,m_2,...,m_d)의 범위에 존재하게 된다. m_i의 값이 Δi (max $_i$ - min $_i$+1)에 근접한다면, 즉 m_i의 값이 커지면 셀의 크기가 작아지기 때문에 이동 객체는 같은 도로 구간을 여러 다른 셀들로 구분하여 표현되어 오버헤드가 많이 발생하게 된다. m_i의 값이 1에 근접한다면, 즉 m_i의 값이 작아지면 셀의 크기가 커져서 셀에 속하게 되는 도로 구간이 많아지게 되므로 경로 탐색이 매우 어렵고 메모리 공간을 많이 차지하게 된다. 그러므로, m_i는 이동 객체의 경로 구간, 즉 도로 구간에 따라서 적절하게 결정하는 것이 바람직하다.

3.1.2 시공간 해시 구조

H-STCA는 시공간 데이타베이스로부터 경로를 탐색하고 이동 객체의 위치 데이타와 속성 정보(이동 객체가 이동하는 도로 교통 정보 등)들을 읽는다. 그리고, 도로 교통에 영향을 미치는 요인에 따른 세부 항목들을 가중치로 분류하여 각 셀들이 참조할 수 있도록 가중치 테이블을 만든다. 또한, 시공간 클러스터링을 위한 임의의 특정 시간에 대한 시공간 해시 구조를 생성한다. H-STCA에 의해 생성되는 시공간 해시 구조는 일반적으로 <그림 6>과 같다. <그림 6>에서 보는 바와 같이 임의의 시간(T_i)에 대한 H-STCA 구조, CELL 구조, RL 구조, DATA 구조, WT 구조로 구성된다.



<그림 6> 시공간 해시 구조

H-STCA 구조는 임의의 시간(T_i)에 대한 시공간 해시 전반에 대한 정보를 저장하는 구조로서 시공간 데이타베이스의 차원의 수는 3차원이고 이동 객체의 개수는 N이다. CELL 구조는 임의의 시간(T_i)에 대한 모든 셀 각각에 대한 정보를 저장하는 구조이다. RL 구조는 특정 셀에 여러 도로가 존재할 경우에 셀 하나를 도로별로 계층화하기 위한 시공간 해시 구조이며 시공간 해시 테이블의 한 셀 안에 2개 이상의 도로 구간이 존재할 경우에는 이를 효율적으로 해결하기 위해 그 셀을 도로 구간으로 계층화(Layer)하였다. 시공간 해시 테이블에서 특정 셀을 분할(Split)하여 처리하는 기존의 방식을 이용하면 그 셀을 분할하고 합병(Merge)하는데 드는 비용은 엄청나게 크다. 이러한 문제를 효과적으로 해결하기 위하여 시공간 해시 테이블을 기존의 해시 구조를 따르지 않고 새로운 확장된 해시 구조를 이용한다.

DATA 구조는 임의의 시간(T_i)에 대한 셀에 소속한 이동 객체들에 대한 데이타를 저장하는데, oid(Object Identifier)는 이동 객체에 대한 참조 포인터이다. WT 구조는 가중치 테이블로서 도로 교통 영향에 미치는 요인들을 분류하고 각 세부 항목별로 가중치를 저장하여 둔 구조이다. 이러한 WT 구조의 예는 <그림 7>과 같다.

분류	세부항목	가중치
도로등급	고속도로	a
	고속화	b
	국도	c
공사	3개월이하	d
	6개월이하	e
	9개월이하	f
	12개월이하	g
	12개월이상	h
차선수	2차선	i
	4차선	j
	6차선	k
	8차선	l
요일	주중	m
	주말	n
	연휴(명절)	o

분류	세부항목	가중치
사고	소형	p
	중형	q
	대형	r
날씨	좋음	s
	보통	t
	나쁨(눈,비,안개등)	u
제한속도	특수(40-50)	v
	저속(60-70)	w
	중속(80-90)	x
	고속(100-110)	y
⋮	⋮	⋮

<그림 7>가중치 테이블 구조

<그림 7>에서 각 세부 항목별로 가중치 값들은 실수로 나타낼 수 있으며 0보다는 크고 1보다는 작거나 같다. 또한, 같은 분류에서는 세부 항목에 따라(교통 흐름에 미치는 영향에 따라) 가중치가 다르며 교통 흐름에 미치는 영향이 큰 세부 항목들이 다른 항목보다 가중치 값이 크다. 이러한 가중치 값들은 사용자에 의해서 더 세분화되어 사용될 수 있으며 다른 값으로 지정되어 사용될 수 있다. 또한, 적용되는 응용 분야에 따라 그 가중치 값을 다르게 지정할 수 있으며 시스템에서 모의실험을 통하여 정의될 수도 있다.

3.2 H-STCA 알고리즘

본 절에서는 H-STCA에서 사용되는 시공간 해시 테이블 생성 알고리즘과 스냅샷 생성의 전처리 과정인 시공간 클러스터링 알고리즘에 대해 상세히 기술한다.

3.2.1 시공간 해시 테이블 생성 알고리즘

시공간 해시 테이블 생성 알고리즘은 <그림 8>과 같다. 입력은 시간 데이타를 나타내는 TIME과 이동 데이타 구조체 및 셀 데이타 구조체를 나타내는 MO와 CELL이 있다. 그리고, 출력은 시공간 해시 테이블 구조체이다. <그림 8>에서 보는 바와 같이 시공간 해시 구조 생성 알고리즘에서 사용되는 함수에는 이동 객체의 위치(position)를 입력하면 셀 ID를 반환하는 FINDCELLID() 함수, 셀 ID(cellID)를 입력하면 셀 포인터를 반환하는 FINDCELL() 함수, 셀 데이타 구조체(c)을 입력하면 셀에 포함되는 도로 정보에 따라 도로별 레이어를 생성하는 ADDLAYER() 함수가 있다. 또한, 셀 데이타 구조체(c)를 시공간 해시 테이블(ht)에 추가하는 ADDCELL() 함수, 이동 객체(mo)를 셀 데이타 구조체(c)에 추가하는 ADDMO() 함수, 그리고 셀 데이타 구조체(c)를 입력하면 해당 셀의 가중치를 반환하는 WEIGHT() 함수가 있다.

```
Algorithm build_spatio-temporal_hash_structure
-----------------------------------------------------------------------------
  Input : TIME, HASH_TABLE, MO, CELL
  Output : 시공간 해시 테이블 구조체
  Method : FINDCELLID(position), FINDCELL(cellID), ADDLAYER(c),
           ADDCELL(ht, c), ADDMO(c, mo), WEIGHT(c)
-----------------------------------------------------------------------------
Begin
   HASH_TABLE *ht
   For each TIME t in CT
     For each MO mo in D
       mo.cellID = FINDCELLID(mo.position);
       c = FINDCELL(mo.cellID); ADDMO(c, mo);  c.count ++;
     end For
     init_min_cells = false;  max_cells = 0;
     For each CELL c in K
       ADDLAYER(c);  c.weight = WEIGHT(c); ADDCELL(ht, c);
       if (c.count > 0 and init_min_cells = false) then
         min_cells = c.count; init_min_cells = true;
       end if
       if (c.count > 0 and init_min_cells = true) then
         if (c.count <= min_cells) then min_cells = c.count;
         end if
         if (c.count > max_cells) then max_cells = c.count;
         end if
       end if
     end For
   end For
   return ht;
End
```

<그림 8> 시공간 해시 테이블 생성 알고리즘

시공간 클러스터링 할 CT 기간 동안 모든 시간 t에 대해서 처리가 완료되면 시공간 클러스터링 알고리즘의 결과로서 시공간 해시 테이블 구조체 포인터 ht가 반환된다. H-STCA는 모든 소속 가능한 셀들에 이동 객체를 삽입하지 않고 실제 이동 객체의 소속 여부를 판정하여 해당되는 셀에 삽입하므로 시공간 클러스터링 시에는 정확한 클러스터인 시공간 해시 테이블들을 발견할 수 있다. 이러한 시공간 해시 테이블들을 추출하여 이동 객체의 경로 탐색을 위한 필요한 지식을 효율적으로 추출할 수 있다. 게다가 시공간 접근 방법으로 시공간 해시를 사용할 때에도 효율성을 높일 수 있다.

3.2.2 시공간 클러스터링 알고리즘

사용자가 시공간 질의 시에 이를 처리하기 위해 시공간 클러스터링을 요구하면 H-STCA는 미리 생성된 시공간 해시 테이블 구조를 사용하여 처리한다. 임계치는 클러스터링을 구분하는 중요한 요소이다. 그러므로, 본 논문에서는 이러한 임계치의 최적 값을 선정하

기 위해 실험을 수행하였고, 그 실험 결과를 통해 최적의 임계치 값을 전체 시공간 해시 테이블의 가중치 합에 따라 10%를 사용하였다.

H-STCA는 임의의 특정 시간에 대한 시공간 해시 테이블의 모든 셀들에 대하여 가중치 테이블을 참조하여 가중치를 구하고, 각 셀들의 가중치를 모두 더한다. 그리고, 가중치 합을 구해서 임계치 미만인 시공간 해시 테이블들을 탐색하여 클러스터를 발견한다. H-STCA에서 임의의 특정 시간에 대해 시공간 해시 테이블의 각 셀 c의 가중치와 모든 셀들의 가중치 합을 구하는 방법은 다음 공식과 같다.

공식 1 (가중치): 특정 시간 T_i에서 $n_j(c)$는 셀 c에 소속된 이동 객체의 개수이고, 셀 c에 적용되는 개별 가중치를 (w_1 (c) , w_2 (c) , ... , w_d (c))이라 할 때, 셀 c의 가중치 $W(c)T_i$는 다음과 같이 계산되어 진다.

$$W(c)Ti = n_j(c) * \sum_{k=1}^{d} w_k(c)$$

(d : 적용될 가중치 개수, $1 \le j \le r$, r : 전체 셀 개수)

공식 2 (가중치 합): 특정 시간 Ti에서 K는 모든 셀들의 집합이며 c_j 가 K에($c_j \in K$)속하고, 셀 c_j의 가중치를 $W(c_j)Ti$ 이라 할 때, 모든 셀 K의 가중치 합 TW(K)Ti 은 다음과 같이 계산되어 진다.

$$TW(K)T_i = \sum_{j=1}^{r} W(c_j)\,T_i \quad (r : 전체\ 셀\ 개수)$$

시공간 클러스터링 알고리즘은 <그림 9>와 같이 입력 부분은 해시 테이블 구조체를 나타내는 HASH_TABLE과 시간 데이타를 나타내는 TIME 및 셀 데이타 구조체를 나타내는 CELL이 있다. 그리고, 출력 부분은 시간별로 전체 셀의 가중치 합이 임계치보다 작은 시공간 해시 테이블들의 배열이다. 게다가 시공간 클러스터링 알고리즘에서 사용되는 함수에는 셀 데이타 구조체(c)를 입력하면 해당 셀의 가중치를 반환하는 WEIGHT() 함수와 시공간 해시 테이블 포인터(ht)에 시공간 해시 테이블(K)을 추가하는 ADDHASH() 함수가 있다.

```
Algorithm spatio-temporal_clustering
  Input : HASH_TABLE, TIME, CELL
  Output : 해시 테이블
  Method : WEIGHT(c), ADDHASH(ht, K)
Begin
   HASH_TABLE  *ht;
   For  each TIME t  in  CT
     total_weight = 0;
     For  each CELL c  in  K
       if  (c.count > 0)  then
          c.weight = WEIGHT(c);
          total_weight = total_weight + c.weight;
       end if
     end For
     if  (total_weight  <  threshold) then
       ADDHASH(ht, K);
     end if
   end For
   return  ht;
End
```

<그림 9> 시공간 클러스터링 알고리즘

시공간 클러스터링할 CT 기간 동안 모든 시간 t에 대해서 처리가 완료되면 시공간 클러스터링 알고리즘의 결과로서 시공간 해시 테이블 포인터 ht가 반환된다.

3.3 지식 추출 알고리즘

본 절에서는 시공간 클러스터링과 스냅샷을 사용한 각각의 지식 추출 알고리즘에 대해 기술한다.

3.3.1 클러스터링 지식

시공간 데이타베이스에 저장된 과거 위치 데이타는

대용량이므로 데이타의 일반적이고 전체적인 경향을 파악하기 위해서 일반화가 필요하다. 클러스터링 지식은 시공간 지식 탐색의 과거 위치 데이타에 대한 일반화에 해당하는 지식이다. 다시 말해 H-STCA의 임계치에 의해 이동 객체의 경로 탐색에 대한 일반적인 경향을 분석하는 것으로서 주어진 임계치에 따라 일반화의 정도가 결정된다.

이동 객체의 과거 위치 데이타 요약 기법인 스냅샷 생성의 전처리 단계인 H-STCA의 시공간 클러스터링에서는 주어진 임계치보다 이동 객체의 개수와 각 셀의 가중치를 이용하여 시공간 해시 테이블의 가중치 합이 낮은 것을 발견한다. 그러나, 이동 객체의 경로 탐색을 위한 지식 추출 단계인 클러스터링 지식 추출에서는 H-STCA을 사용하여 발견된 시공간 해시 테이블들의 클러스터뿐만 아니라 H-STCA에서는 무시되었던 시공간 해시 테이블들도 추출한다. 왜냐하면 경우에 따라서는 일반적인 경향을 따르지 않는 데이타가 더욱 유용하기 때문이다.

예를 들어, "2004년 8월(1일~31일)에 서울에서 부산까지 가는데 교통이 원활했던 날은 언제인가?" 라는 시공간 질의뿐만 아니라, "2004년 8월(1일~31일)에 서울에서 부산까지 가는데 교통이 혼잡했던 날은 언제인가?" 라는 시공간 질의에 대한 수행을 처리하기 위해서는 시공간 클러스터링 지식의 예외로서 임계치보다 시공간 해시 테이블의 가중치 합이 높은 것을 발견하는 것이 의사 결정을 위해 필요할 수도 있기 때문이다. 그러므로, 임계치에 따라 임계치 미만인 시공간 해시 테이블과 임계치 이상인 시공간 해시 테이블로 구분하여 시공간 해시 테이블을 추출한다. 시공간 클러스터링을 통한 클러스터링 지식 추출 알고리즘은 <그림 10>과 같다.

클러스터링 지식 추출 알고리즘에서 클러스터링 지식 추출을 위한 입력 부분은 해시 테이블 구조체를 나타내는 HASH_TABLE과 시간 데이타를 나타내는 TIME 및 셀 데이타 구조체를 나타내는 CELL이 있다. 그리고, 출력 부분은 시간별로 전체 셀의 가중치 합이 임계치보다 작은 시공간 해시 테이블들의 시간을 추출하는 배열과 전체 셀의 가중치 합이 임계치보다 같거나 큰 시공간 해시 테이블들의 시간을 추출하는 배열이다.

클러스터링 지식 추출 알고리즘에서 사용되는 함수에는 해시 테이블 포인터(h_table)를 입력하면 해당 해시 테이블의 가중치 합을 반환하는 T_WEIGHT() 함수와 분류된 시공간 해시 테이블들의 시간을 추출하기 위해서 시간 포인터(tp)에 시간(t)를 추가하는 ADDTIME() 함수가 있다.

```
Algorithm clustering_knowledge_extraction
Input : HASH_TABLE, TIME, CELL
Output : 시간 클러스터링
Method : T_WEIGHT(h_table), ADDTIME(tp, t)
Begin
  TIME  *low_tp, *high_tp;
  HASH_TABLE  *ht
  For  each TIME t  in  CT
    For  each HASH_TABLE h_table  in  HT
     if  (h_table.count > 0)  then
         h_table.weight = T_WEIGHT(h_table);
     if  ( h_table.weight <  threshold)  then
        ADDTIME(low_tp, t);
     else if
         ADDTIME(high_tp, t);
     end if
  end For
  return  low_tp;
  return  high_tp;
End
```

<그림 10> 클러스터링 지식 추출 알고리즘

시공간 클러스터링할 CT 기간 동안 모든 시간 t에 대해서 처리가 완료되면 클러스터링 지식 추출 알고리즘의 결과로서 시간 배열에 대한 시간 포인터 low_tp, high_tpt가 반환된다.

3.3.2 스냅샷 지식

스냅샷 지식은 과거 위치 데이타 마이닝의 과거 위치 데이타에 대한 요약에 해당하는 지식이며 스냅샷의 최적 경로 값의 합에 따른 시간 지식으로써 스냅샷의 최적 경로 값의 합이 가장 작은 스냅샷의 시간으로 추출된다. 예를 들면, "2004년 8월(1일~31일)에 서울에서 부산까지 가는데 교통이 원활했던 날은 언제인가?"를 파악하기 위해서는 먼저 시공간 클러스터링을 사용하여 전처리하고, 그 후에 클러스터링 결과를 갖고 스냅샷을 만들어 시간 지식을 추출한다.

H-STCA의 시공간 클러스터링에서 주어진 임계치보다 이동 객체의 개수와 각 셀의 가중치를 이용하여 시공간 해시 테이블의 가중치 합이 임계치보다 작은 것을 발견한 시공간 해시 테이블들을 대상으로 모든 시공간 해시 테이블들의 각 셀들의 도로 구간 거리 및 그 도로 구간에서의 이동 객체의 표준 평균 속력을 계산한다. 그리고, 그 도로 구간을 표준 평균 속력으로 나누어 표준 평균 소요 시간을 구한다. 그 다음에 시공간 해시 테이블들의 각 셀들의 최적 경로 값을 구하여 스냅샷을 생성한다. 최적 경로 값은 표준 평균 소요 시간에 셀 가중치를 곱한 값이므로 생성된 모든 스냅샷에서 각 셀들의 도로 구간에 대한 최적 경로 값의 합을 모두 구한다. 이러한 최적 경로 값의 합을 모두 비교하여 최적 경로 값의 합이 가장 작은 스냅샷의 시간 지식을 추출한다. 이러한 스냅샷 지식 추출 알고리즘은 <그림 11>과 같다.

스냅샷 지식 추출 알고리즘은 <그림 11>과 같이 스냅샷 지식 추출을 위한 입력 부분은 해시 테이블 구조체를 나타내는 HASH_TABLE과 스냅샷 구조체를 나타내는 SNAPSHOT이 있다. 그리고, 시간 데이타를 나타내는 TIME 및 셀 데이타 구조체를 나타내는 CELL이 있다. 출력 부분에서는 시공간 클러스터링에 의해 구해진 시공간 해시 테이블들을 토대로 시간별로 전체 셀의 최적 경로 값의 합이 가장 작은 스냅샷의 생성 시간이 추출된다.

스냅샷 지식 추출 알고리즘에서 사용되는 함수에는 해시 테이블 포인터(h_t)를 입력하면 해당 해시 테이블의 가중치 합을 반환 T_WEIGHT() 함수와 시공간 해시 테이블 포인터(ht)에 시공간 해시 테이블(K)을 추가하는 ADDHASH() 함수가 있다. 그리고, 셀 데이타 구조체(c.r)를 입력하면 해당 셀에 포함된 도로 구간의 거리를 계산하여 반환하는 ROADDIST() 함수와 셀 데이타 구조체(c.r)를 입력하면 해당 셀에 포함된 도로 구간의 표준 평균 속도를 계산하여 반환하는 MOSPEED() 함수가 있다. 게다가 해시 테이블 포인터(ht)에 포함된 해시 테이블 중 최적 경로 값이 가장 작은 해시 테이블의 생성 시간을 반환하는 MINOP() 함수가 있다.

```
Algorithm snapshot_knowledge_extraction
Input : HASH_TABLE, TIME, CELL
Output : 최적 시간 및 경로를 가지는 해시 테이블 구조체 포인터
Method : T_WEIGHT(h_t), ADDHASH(ht, K), ROADDIST(c.r), MOSPEED(c.r), MINOP(ht)
Begin
  HASH_TABLE  *ht, best_ht;
  For  each TIME t  in  CT
    For  each HASH_TABLE h_t  in  HT
     if  (h_t.count > 0) then h_t.weight = T_WEIGHT(h_t);
     end if
    end for
    if  ( h_t.weight <  threshold) then ADDHASH(ht, K);
    end if
  end For
  For each HASH_TABLE h in ht
      For each CELL c in h
        dist = ROADDIST(c.r); speed = MOSPEED(c.r);
        need_time = dist / speed; optimal_path = c.weight * need_time;
        total_op = total_op + optimal_path;
      end For
      h.total_op = total_op; total_op = 0; best_ht = MINOP(ht);
    end For
    return best_ht;
End
```

<그림 11> 스냅샷 지식 추출 알고리즘

모든 스냅샷에 대해서 처리가 완료되면 스냅샷 지식 추출 알고리즘의 결과로서 최적 경로의 값의 합이 가장 작은 스냅샷 생성 시간 best_ht가 반환된다.

4. 성능 평가

본 장에서는 H-STCA의 최적 임계치를 선정하기 위한 실험을 수행하였고, 또한 H-STCA의 성능 평가를 위해서 기존 스냅샷 방법 및 기존의 시공간 인덱스 기법과 비교 실험을 수행하였다.

4.1 실험 환경

본 논문에서 성능 평가를 위한 실험의 환경은 다음과 같다. 실험에 사용된 시스템 사양은 Intel Pentium 4 1.7GHz 프로세서와 512MB 메모리를 사용하였다. 그리고, 운영체제는 Redhat Linux 9.0을 사용하였다. 실험에 사용된 이동 객체 데이타 집합은 네트워크 기반 위치 데이타 생성기를 통해 생성하였고, 이동 객체 데이타는 1개월 당 10만 건의 위치 데이타를 생성하였다. 전체 기간을 10개월로 가정하여 100만개의 이동 객체 데이타를 생성하였으며 시공간 해시 테이블은 시간 당 1개가 생성되도록 하였다. 이동 객체 데이타는 국내 고속도로를 기반으로 서울에서 부산까지 가는데 6개의 도로 경로에 대해 고속도로 상에서 이동하는 이동 객체의 위치 데이타를 생성하였다.

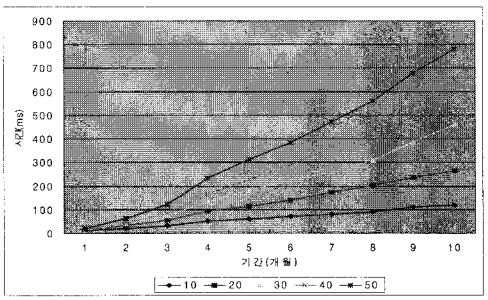
성능 평가를 위한 시공간 질의 예는 다음과 같다. 약 10개월 이동 객체의 과거 위치 데이타를 이용하여 최적 경로 값을 계산하고 교통이 원활한 이동 객체의 경로 탐색을 위한 지식을 추출하기 위하여 본 논문에서 제안한 지식 추출 알고리즘과 H-STCA를 사용하였다. 그리고, 이러한 시공간 질의는 임의의 1개월에서 10개월 사이의 시간 간격에 따라 "특정 목적지(서울~부산)로 휴가를 가려고 하는데 언제(몇월 몇일 몇시)(교통이 가장 원활한 시간), 어느 길로(최적 경로) 가야 하는가?"라는 시공간 질의를 사용하였다.

4.2 H-STCA 임계치 실험

본 절에서는 H-STCA 임계치를 전체 시공간 해시 테이블의 가중치 합에 따라 10%, 20%, 30%, 40%, 50%로 주어 비교 실험을 수행하여 최적의 임계치 값을 선정하였다. 또한, 실험에 사용되는 시공간 질의 예에서 대용량의 이동 객체 데이타에 대한 검색 시간과 최적 경로 탐색 시간을 비교하였다.

4.2.1 이동 객체 검색 시간 실험

<그림 12>는 이동 객체 검색 시간에서 비교 실험 결과를 나타낸 그래프이다. <그림 12>에서 보는 바와 같이 H-STCA에서 임계치가 10%일 때의 검색 시간이 임계치가 20%~50%일 때 보다 약 2배~5배 이상 성능이 좋은 것으로 나타났다. 또한, 이동 객체 데이타 개수가 증가함에 따라 이동 객체 검색 시간이 H-STCA에서 임계치가 20%~50%일 때 임계치가 10%일 때 보다 급격히 증가함을 알 수 있다. 그러므로, H-STCA의 최적 임계치 값은 전체 시공간 해시 테이블의 가중치 합에 따라 10%로 선정하여 수행하는 것이 바람직하다는 결론을 얻을 수 있다.

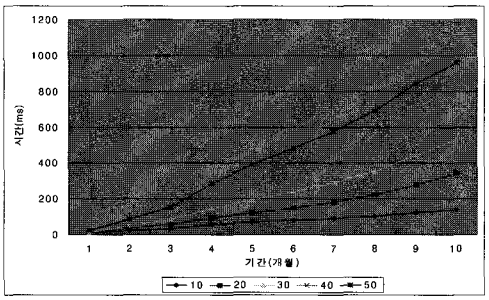


<그림 12> 이동 객체 검색 시간 비교 : 임계치 10%~50%

4.2.2 최적 경로 탐색 시간 실험

<그림 13>은 이동 객체 최적 경로 탐색 시간에서 비교 실험 결과를 나타낸 그래프이다. <그림 13>에서 보는 바와 같이 H-STCA에서 임계치가 10%일 때의 이동 객체 최적 경로 탐색 시간이 임계치가 20%~50%일 때 보다 약 2.5배~6배 이상 성능이 좋은 것으로 나타났다. 또한, 이동 객체 데이타 개수가 증가함에 따라 이동 객체 최적 경로 탐색 시간이 H-STCA에서 임계치가 20%~50%일 때 임계치가 10%일 때 보다 급격히 증가함을 알 수 있다. 그리고, H-STCA에서 임계치가 50%이상인 경우는 이동 객체 최적 경로 탐색 시간이 H-STCA에서 임계치가 10%일때 보다 현저히 높음을 알 수 있다. 이러한 결과를 비추어 볼때 이동 객체 데이타 개수가 많으면 많을수록 이동 객체 최적 경로 탐색 시간은 더욱 더 증가한다. 그러므로, H-

STCA의 최적 임계치 값은 전체 시공간 해시 테이블의 가중치 합에 따라 10%로 선정하여 수행하는 것이 가장 바람직하다는 결론을 얻을 수 있다.



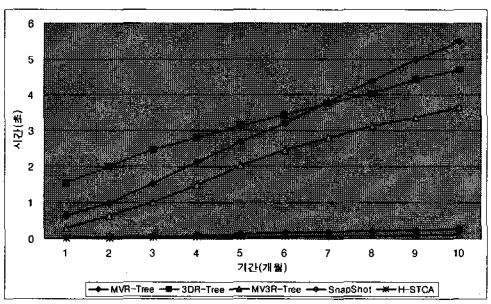
<그림 13> 이동 객체 최적 경로 탐색 시간 비교 : 임계치 10%~50%

4.3 H-STCA 성능 평가

본 절에서는 H-STCA의 성능 평가를 위한 기존 스냅샷 방법과 기존의 시공간 인덱스 기법과의 비교 실험을 위해서 H-STCA에서의 임계치는 4.2절에서 실험에 의해 산출된 전체 10%로 선정하여 적용하였다.

4.3.1 이동 객체 검색 시간 실험

<그림 14>는 기존 시공간 인덱스와 스냅샷과의 이동 객체 검색 시간의 실험 결과를 비교한 그래프이다. 대용량의 이동 객체 데이타에 대한 검색에서 시공간 인덱스 및 스냅샷을 사용하는 비용이 H-STCA을 사용하는 스냅샷 클러스터링 비용보다 매우 높다는 것을 보여준다. 10만개에서 100만개의 데이타에 대하여 H-STCA를 사용하는 스냅샷 클러스터링 비용과 시공간 인덱스를 사용한 비용의 차이는 약 30배에서 65배까지 나타났고 스냅샷을 사용한 검색보다 약 2배에서 3배의 성능 향상을 보였다.

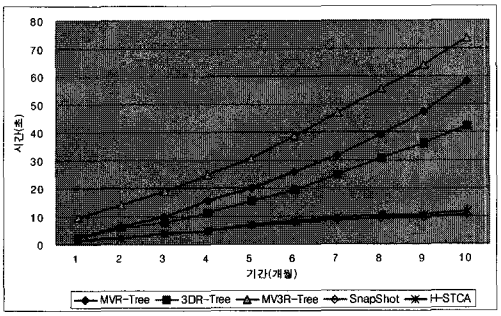


<그림 14> 검색 시간 성능 비교

4.3.2 이동 객체 저장 구조 생성 시간 실험

<그림 15>는 기존 시공간 인덱스와 스냅샷과의 저장 구조 생성 시간의 실험 결과를 비교한 그래프이다. 대용량의 이동 객체 데이타에 대한 저장 구조 생성에서 시공간 인덱스의 저장 구조를 생성하는 시간이 H-STCA를 사용한 스냅샷 클러스터링 방법의 저장 구조를 생성하는 시간보다 높다는 것을 보여준다. 10만개에서 100만개의 데이타에 대하여 H-STCA를 사용한 스냅샷 클러스터링 방법의 저장 구조 생성 시간과 시공간 인덱스의 저장 구조 생성 시간의 차이는 약 2.5배에서 5.5배까지 나타났다.

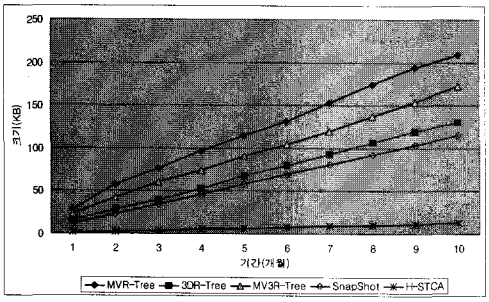
그러나, H-STCA를 사용한 스냅샷 클러스터링 방법의 저장 구조 생성 시간이 스냅샷을 사용한 저장 구조 생성 시간보다 약 1.1배의 성능이 떨어진다. 이는 H-STCA가 전체 이동 객체 데이타 집합을 일정한 기간 동안 클러스터링을 수행하는 시간이 포함되어 H-STCA를 사용한 스냅샷 클러스터링 방법의 저장 구조를 생성하는 시간이 증가하기 때문이다.



<그림 15> 저장 구조 생성 시간 성능 비교

4.3.3 이동 객체 저장 공간 생성 크기 실험

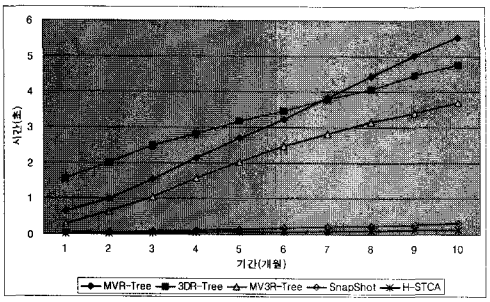
<그림 16>은 기존 시공간 인덱스와 스냅샷과의 저장 공간 생성 크기의 실험 결과를 비교한 그래프이다. 대용량의 이동 객체 데이타에 대한 저장 공간 생성 크기에서 시공간 인덱스 및 스냅샷을 사용하는 저장 공간 생성 크기 비용이 H-STCA를 사용한 스냅샷 클러스터링 방법을 사용하는 비용보다 매우 높다는 것을 보여준다. 10만개에서 100만개의 데이타에 대하여 H-STCA를 사용한 스냅샷 클러스터링 방법을 사용하는 비용과 시공간 인덱스를 사용한 비용의 차이는 약 10배에서 20배까지 나타났고 스냅샷을 이용한 저장 공간 생성 크기 비용보다 약 10배의 성능 향상을 보였다.



<그림 16> 저장 공간 생성 크기 성능 비교

4.3.4 이동 객체 최적 경로 탐색 성능 비교 실험

<그림 17>은 기존 시공간 인덱스와 스냅샷과의 최적 경로 탐색 성능 비교에서 H-STCA를 사용한 스냅샷 클러스터링 방법을 응용하여 최적 경로 탐색의 실험 결과를 비교한 그래프이다. 대용량의 이동 객체 데이타에 대한 최적 경로 탐색 성능에서 시공간 인덱스 및 스냅샷을 사용하여 최적 경로 탐색 비용이 H-STCA를 사용한 스냅샷 클러스터링 방법을 사용하는 비용보다 매우 높다는 것을 보여준다. 10만개에서 100만개의 데이타에 대하여 H-STCA를 사용한 스냅샷 클러스터링 방법을 사용하는 최적 경로 탐색 비용과 시공간 인덱스를 사용한 최적 경로 탐색 비용의 차이는 약 30배에서 45배까지 나타났고 스냅샷을 이용한 최적 경로 탐색 비용보다 약 2.3배의 성능 향상을 보였다.



<그림 17> 최적 경로 탐색 시간 성능 비교

5. 결 론

본 논문에서는 시공간 질의 처리 시에 기존 시공간 인덱스 기법과 스냅샷 기법의 단점을 해결하기 위하여 기존 공간 클러스터링 기법을 시간 축에 대한 시공간 클러스터링으로 확장하였다. 또한, 이를 이용하여 스냅샷 생성의 전처리 과정으로 해시-기반 시공간 클러스터링 알고리즘인 H-STCA를 제안하였다. 또한, 제안한 해시-기반 시공간 클러스터링 알고리즘인 H-STCA를 사용하여 과거 위치 데이타로부터 이동 객체의 경로 탐색을 위한 지식을 추출하기 위한 지식 추출 알고리즘을 개발하였다.

본 논문에서 제안한 H-STCA를 사용한 스냅샷 클러스터링 방법에 대한 성능 평가로 기존 과거 위치 인덱스 기법을 사용한 방법 및 스냅샷만을 만들어 사용하는 방법과 실험을 통하여 성능을 평가한 결과는 다음과 같다. 첫째, 대용량의 이동 객체 데이타에 대한 이동 객체 검색 시간에서 H-STCA를 사용한 스냅샷 클러스터링 방법이 시공간 인덱스를 사용하는 것 보다 약 30배에서 65배 이상의 성능을 보였고, H-STCA를 사용한 스냅샷 클러스터링 방법은 스냅샷 방법보다 약 2배에서 3배의 성능 향상을 보였다. 둘째, 대용량의 이동 객체 데이타에 대한 저장 구조 생성 시간에서 H-STCA를 사용한 스냅샷 클러스터링 방법이 기존 인덱스 방법보다 약 2.5배에서 5.5배 이상의 성능이 우수하였으나 스냅샷 방법보다는 약 1.1배 성능이 떨어졌다.

셋째, 저장 구조 생성 시간에서는 H-STCA를 사용한 스냅샷 클러스터링 방법이 시공간 인덱스를 사용한 저장 구조 생성 시간보다 약 10배에서 20배 이상 성능 향상을 보였고, 스냅샷을 이용한 방법보다는 약 10배의 성능 향상을 보였다. 마지막으로, 최적 경로 탐색의 실험 결과는 H-STCA를 사용한 스냅샷 클러스터링 방법의 성능이 기존 인덱스 방법보다는 약 30배에서 45배 이상 우수하였으며, 스냅샷보다는 약 2~3배의 성능 향상을 보였다. 특히, 대용량의 이동 객체 데이타를 이용한 최적 경로 탐색에서도 이동 객체 데이타에 대한 I/O를 크게 줄이면서 매우 높은 성능을 보여주었다.

본 논문의 향후 연구 과제는 다음과 같다. 본 연구에서 제안하여 개발한 알고리즘인 H-STCA를 실제 적용한 시스템의 구현이 필요하다. 또한, 도로 교통 정보에 따른 도로 여건 및 교통적 여건에 따라 이동 객체의 도로 구간에서 흐름 평가의 판단 기준이 될 수 있는 가중치 선정 모델의 연구가 필요하다. 그리고, H-STCA는 이동 객체를 이동 점으로 간주하여 처리하고 있으므로, 이동 객체의 이동 궤적을 효율적으로 처리할 수 있도록 하는 시공간 데이타 모델을 연구하여 H-STCA의 알고리즘을 확장할 필요가 있다.

참고문헌

[1] Sistla, A. P., Wolfson, O., Chamberlain, S., and Dao, S., "Modeling and Querying Moving Objects," Proc. of International Conference on Knowledge and Data Engineering, 1997, pp.422-432.

[2] 윤재관, 한기준, "LBS(Location Based Service)를 위한 기술 개발 동향," 대한전자공학회 전자공학회지, Vol.29, No.12, 2002, pp.1425-1434.

[3] 이기영, 윤재관, 한기준, "HBR-tree를 이용한 실시간 모바일 GIS의 개발," 개방형지리정보시스템학회 논문지, Vol.6, No.1, 2004, pp.73-85.

[4] Yuan, M., Temporal GIS and Spatio-Temporal Modeling, NCGIA, SANTA_FE_CD-ROM, 1996.

[5] Yuan, M., "Use of a Three-Domain Representation to Enhance GIS Support for Complex Spatio-Temporal Queries," IEEE Transaction on Geographic Information System, Vol.3, No.2, 1997, pp.137-159.

[6] 장인성, 이기준, "밀도를 이용한 K-최근접 탐색방법," 한국정보과학회 논문지, Vol.27, No.2, 2000, pp.80-82.

[7] Pfoser, D., Jensen, C. S., and Theodoridis, Y., "Novel Approaches in Query Processing for Moving Objects," Proc. of International Conference on VLDB, 2000, pp.395-406.

[8] Hadjieleftheriou, M., Kriakov, V., Tao, Y., Kollios, G., Delis, A., and Tsotras, V., "Spatio-Temporal Data Services in a Shared-Nothing Environment," Proc. of International Conference on Scientific and Statistical Database Management, 2004, pp.131-139.

[9] Tao, Y., and Papadias, D., "MV3R-Tree: A Spatio-Temporal Access Method for Timestamp and Interval Queries," Proc. of International Conference on VLDB, 2001, pp.431-440.

[10] Guting, R. H., Erwig, M., Jensen, C. S., Lorentzos, N. A., Schneider, M., and Vazirgiannis, M. "A Foundation for Representing and Querying Moving Objects," ACM Transaction on Database Systems, Vol. 25, No.1, 2000, pp.1-42.

[11] Pfoser, D., Theodoridis, Y., and Jensen, C. S., Indexing Trajectories of Moving Point Objects, Chorochronos Technical Report, 1999.

[12] Wolfson, O., Chamberlain, S., Sistal, P., Xu, B., and Zhon, X., "DOMINO: Databases for Moving Objects Tracking," Proc. of ACM SIGMOD Conference, 1999, pp.547-549.

[13] 개방형지리정보시스템학회, 대용량 위치정보 관리 컴포넌트 기술 개발, 한국전자통신연구원 최종보고서, 2003.

이기영

1984년 숭실대학교 전자계산학과 졸업 (공학사)

1984년~1990년 한국해양연구원(KORDI) (연구원)

1988년 건국대학교 컴퓨터공학과 졸업(공학석사)

2005년 건국대학교 컴퓨터공학과 졸업(공학박사)

1991년 ~ 현재 서울보건대학 인터넷정보과 부교수

관심분야 : GIS, LBS, 데이타마이닝, 위치 데이타 인덱스, 텔레매틱스

강홍구

2002년 건국대학교 컴퓨터공학과 졸업 (공학사)

2004년 건국대학교 대학원 컴퓨터공학과 졸업(공학석사)

2004년 ~ 현재 건국대학교 대학원 컴퓨터공학과 박사과정

관심분야 : GIS, 공간데이타베이스, MMDBMS, MODB, LBS

윤재관

1997년 건국대학교 컴퓨터공학과 졸업 (공학사)

1999년 건국대학교 컴퓨터공학과 졸업 (공학석사)

2003년 건국대학교 컴퓨터공학과 졸업(공학박사)

2003년 ~ 현재 건국대학교 컴퓨터공학부 강의교수

관심분야 : 실시간 데이타베이스, LBS, 위치 데이타 인덱스, 분산 위치 저장 시스템

한기준

1979년 서울대학교 수학교육학과 졸업 (이학사)

1981년 한국과학기술원(KAIST) 전산학과 졸업 (공학석사)

1985년 한국과학기술원(KAIST) 전산학과 졸업 (공학박사)

1990년 Stanford 대학 전산학과 visiting scholar

1985년~현재 건국대학교 컴퓨터공학부 교수

2004년~현재 한국공간정보시스템학회 회장

2004년~현재 한국정보시스템감리사협회 회장

관심분야 : 공간데이타베이스, GIS, LBS, 텔레매틱스, 정보시스템 감리