韓國 컴퓨터情報學會 論文誌
第12卷 第2號, 2007. 5.

2007-12-2-2-8

R*-Tree와 Grid를 이용한 이동 객체의 위치 일반화 기법

고 현*, 김 광 종**, 이 연 식***

Location Generalization Method of Moving Object using R*-Tree and Grid

Ko Hyun *, Kim Kwang Jong **, Lee Yon Sik ***

요 약

패턴 탐사에 관한 기존의 연구들[1,2,3,4,5,6,11,12,13]은 이동 객체의 위치 이력 데이터 집합에 대한 위치 일반화 접근법을 사용하지 않거나 사용해도 특정 공간상의 이동 패턴들 중 단순히 시공간 제약이 없는 빈발 패턴만을 추출하므로, 특정 지점들 간의 최적 이동 경로나 스케줄링 경로와 같은 시공간 제약을 갖는 빈발 패턴 탐사에는 적용하기 어렵다. 또한 패턴 탐사의 수행에 있어 기존의 기법들은 데이터베이스에 대한 반복 접근을 줄이기 위해 메모리 상에 패턴 트리를 생성하여 사용하므로 보다 많은 메모리 공간을 소요하게 된다. 따라서 이러한 기존 탐사 기법들의 문제점들을 해결하기 위한 보다 효율적인 패턴 탐사 기법이 필요한 실정이다. 효율적 탐사 기법을 개발하기 위하여 본 논문에서는 방대한 이동 객체의 이력 데이터 집합에 대한 탐사 수행 시간 및 탐사에 필요한 메모리 공간을 최소화하기 위해서 상세 수준의 데이터들을 의미있는 공간영역 정보로 변환하는 새로운 위치 일반화 방법을 제안한다. 제안된 방법은 패턴 탐사의 전처리 과정에서 R*-Tree와 영역 Grid 해쉬 테이블(AGHT:Area Grid Hash Table)을 기반으로 이동 객체의 위치 속성들을 2차원 공간영역으로 일반화하여 이동 시퀀스를 생성함으로써 효율적인 이동 객체의 공간 이동 패턴 마이닝을 유도할 수 있다.

Abstract

The existing pattern mining methods[1,2,3,4,5,6,11,12,13] do not use location generalization method on the set of location history data of moving object, but even so they simply do extract only frequent patterns which have no spatio-temporal constraint in moving patterns on specific space. Therefore, it is difficult for those methods to apply to frequent pattern mining which has spatio-temporal constraint such as optimal moving or scheduling paths among the specific points. And also, those methods are required more large memory space due to using pattern tree on memory for reducing repeated scan database. Therefore, more effective pattern mining technique is required for solving these problems. In this paper, in order to develop more effective pattern mining

• 제1저자 : 고 현
• 접수일 : 2007.4.16, 심사일 : 2007.4.23, 심사완료일 : 2007. 5.16.
* 군산대학교 컴퓨터정보과학과 박사과정, ** 군산대학교 컴퓨터정보과학과 이학박사
*** 군산대학교 컴퓨터정보과학과 교수
※ 이 논문은 한국과학재단 특정기초연구(R01-2004-000-10946-0)지원으로 수행되었음

technique, we propose new location generalization method that converts data of detailed level into meaningful spatial information for reducing the processing time for pattern mining of a massive history data set of moving object and space saving. The proposed method can lead the efficient spatial moving pattern mining of moving object using by creating moving sequences through generalizing the location attributes of moving object into 2D spatial area based on R*-Tree and Area Grid Hash Table(AGHT) in preprocessing stage of pattern mining.

▸ Keyword : 이동 객체(Moving Object), 이동 패턴 탐사(Moving Pattern Mining), 위치 일반화(Location Generalization), 시공간 데이터 마이닝(Temporal Data Mining)

Ⅰ. 서 론

최근 이동 객체의 위치 추적 및 무선통신 기술의 발달로 휴대폰, 자동차 등과 같이 시간의 흐름에 따라 공간 및 비공간 속성이 끊임없이 변화하는 시공간 객체들의 위치 정보를 이용하여 교통관제나 텔레매틱스, 기상예측, 마케팅 등과 같이 다양한 분야에서 활용할 수 있는 새로운 서비스를 개발하기 위한 노력들이 활발히 진행되고 있다. 시공간 데이터 마이닝은 시간 및 공간, 시공간 특성을 포함하고 있는 방대한 시공간 데이터 집합으로부터 이전에 알려지지 않았던 잠재적으로 유용한 지식을 탐사하여 새로운 응용 서비스를 개발하기 위한 기술로 현재까지 많은 연구들이 수행되었다. 선행 연구들 중 시공간 패턴 탐사 방법들[1,2,3,4,5,6,11,12,13]은 시간 순서에 따라 발생하는 이벤트나 시간 변화에 따라 규칙적이고 반복적으로 발생되는 이벤트들의 패턴을 탐사하기 위한 기법으로, 시계열 데이터나 시퀀스 데이터 분석에 이용될 수 있다. 현재까지의 패턴 탐사 기법들로는 STPMine1-2[1], GSP[2], MP[3,4], MPMine[11, 12], STMPE[13] 등이 있으며, 이러한 기법들 중 일부는 연관규칙인 Apriori 계열의 알고리즘을 변형하여 패턴 탐사를 수행한다.

이동 객체의 연속적인 이동 위치는 이산적인 시점에서 샘플링 된다고 가정할 때 평면상의 (x,y) 좌표 형태로 표현되기 때문에 이러한 상세 수준의 위치 정보로부터 의미있는 패턴을 찾는 것은 매우 어려운 일이다. 따라서 시공간 패턴 탐사 기법들은 이동 객체의 위치값과 각 위치에서의 유효시간을 사용자 정의에 따른 공간 개념 계층이나 시간 개념 계층의 일반화된 값으로 변환하기 위해서 시공간 일반화 연산을 사용한다. 하지만 기존의 패턴 탐사에서 적용하는 일반화 방법들은 단순히 공간 영역이나 시간 영역에 대한 추상 개념 계층을 기반으로 일반화를 수행하기 때문에 최하위 레벨의 공간 및 시간 영역 내에서의 이동 패턴까지 고려해야 하는 경우에는 적합하지 않다. 즉, 공간상의 두 지점들에 대한 최적 이동 궤적 탐색 질의 처리 시 단순히 의미있는 수준의 공간 영역 시퀀스로 궤적을 표현하기 때문에, 각 영역 내에서의 이동 이력이 모호하게 되어 부정확한 이동 궤적을 제공할 수 있음으로 최하위 레벨의 공간 영역일지라도 영역 내 패턴 분석이 반드시 필요하다.

이에 본 논문에서는 이동 객체의 위치값과 공간 영역 간의 위상 관계를 고려하여 레벨별 공간 영역과 이동 경로 구간의 분기점인 기점노드로 구성된 추상 공간 개념 계층을 기반으로 상세 수준의 위치 이력 데이터들을 의미있는 공간영역으로 일반화하는 방법을 제시한다. 제안된 방법은 R*-Tree와 AGHT를 기반으로 이동 객체의 위치 속성을 2차원 공간영역으로 일반화함으로써 이동 객체의 이력 데이터 집합에 대한 패턴 탐사 수행 시간 및 탐사에 필요한 메모리 공간을 최소화함으로써 효율적인 패턴 탐사를 수행할 수 있다.

본 논문의 구성은 다음과 같다. 2장에서는 관련연구로 기존의 이동 객체 패턴 탐사 기법에 대해 기술하고, 3장에서는 이동 객체와 공간 개념 계층에 대해 정의한다. 4장에서는 위치 일반화를 위한 공간 연산 알고리즘을 제안하고 5장에서는 제안한 알고리즘을 구현한다. 마지막 6장에서는 결론 및 향후 연구 과제를 제시한다.

Ⅱ. 관련연구

시공간 데이터 마이닝은 시공간 특성을 포함하고 있는 방대한 시공간 데이터 집합으로부터 묵시적이고 잠재적으로 유용한 지식을 추출하는 기술로서, 시공간 객체들의 공통적이고 일반적인 성질을 이용한 객체들의 집단화, 시공간 객체들 간의 시간 및 공간, 시공간 상에서의 연관 관계 분석, 시공간 상에서의 변화에 대한 패턴 및 예측과 같은 것을 목적으로 한다[12]. 현재까지 시공간 데이터 마이닝에 대한 많은 연구

가 수행되었으며, 특히 기존의 연구들 중 공간상에서의 다양한 객체의 이동변화로부터 의미있는 패턴을 추출하기 위한 시공간 패턴 탐사 기법들[1,2,3,4,5,6,11,12,13]은 교통관제나 텔레매틱스, 기상예측, 마케팅 등 다양한 분야로 위치기반 서비스를 접목시키기 위한 마이닝 기법 중의 하나이다.

데이터 마이닝 기법들 중 일반적인 데이터에 존재하는 패턴을 탐사하기 위한 기법들로 순차 패턴[7,8,9], 주기 패턴[10] 등이 있다. 이러한 기법들은 시간 및 공간 속성, 시공간 속성에 대한 고려를 통해 시공간 데이터의 패턴 추출을 위한 여러 기법들의 기반이 되었다. 현재까지 연구된 시공간 패턴 탐사 기법들 중 Apriori 계열의 알고리즘을 변형한 STPMine1[1]은 이동 객체의 이력 데이터로부터 주기적 패턴을 추출하는 기법으로, 패턴 탐사 시 데이터베이스를 반복해서 스캔하는 문제가 있다. STPMine2[1]에서는 이를 해결하기 위해서 빈발 1-패턴을 조합한 Super와 Shadow 패턴을 생성, Max_Subpattern 트리를 구성하여 빈발 패턴을 추출함으로써 데이터베이스의 스캔 횟수를 크게 줄이는 방법을 사용하였다. 그러나 Super와 Shadow 패턴의 최소지지도가 낮거나 또는 이동 객체 수가 증가하거나, 시간 영역의 수가 늘어날수록 후보 패턴의 수가 증가하여 비교 연산 횟수 및 소요 메모리 량이 증가하는 단점이 있다.

STPMine1-2[1]와 마찬가지로 GSP[2]도 Apriori 알고리즘에 기반한 순차 패턴 추출 알고리즘으로, 시간 제한(time constraints), 이동 윈도우(sliding time windows), 분류(taxonomies) 등을 도입하여 순차 패턴 추출을 일반화하지만, 많은 데이터베이스 스캔 시간이 필요하고 긴 순차 패턴의 탐사 시 성능이 저하되는 문제가 있다. STMPE[13]는 이동 객체의 이력 정보를 분석하여 시공간 정보를 포함하는 일반화된 이동 패턴을 추출하는 기법으로, 시공간 데이터 일반화, 이동 패턴 추출, 이동 패턴 트리 생성 기능을 지원하여 데이터베이스의 스캔 횟수를 줄이고 패턴 탐사 수행시간을 최소화할 수는 있지만 이동 패턴 트리의 크기가 커질수록 많은 메모리 공간을 소요하게 된다. 마지막으로 MP[3,4]와 MPMine[11,12]은 전처리 과정에서 이동 객체의 이력 데이터 집합으로부터 위치 요약 및 일반화를 통해 탐색공간을 줄일 수 있는 효율적인 패턴 탐사 기법이지만, 규칙적인 시간의 변화에 따른 시간 정보가 없는 이동 패턴만을 추출하는 문제가 있다. 지금까지 고찰한 시공간 패턴 탐색 기법들은 단순히 시간 및 공간에 대한 제약이 없는 빈발 패턴만을 추출하기 때문에 특정 지점들 간의 최적 이동 경로나 특정 시간기간 동안의 최적 스케줄링 경로와 같이 시간 및 공간 제약을 갖는 빈발 패턴을 탐사하는 문제에는 적용하기 어렵다.

지금까지의 패턴 탐색 기법들 중 일부는 객체의 이력 데이터에 대한 일반화 접근법을 사용하여 패턴 탐사를 수행하였다. STMPE[13]는 이동 객체의 시간 속성과 공간 속성에 대해 각각 시간 및 공간 개념 계층으로의 일반화를 수행하였으나, 실세계의 시간 및 공간 개념에 대한 고려없이 단순히 마이닝 대상 데이터 자체를 사용자가 정의한 특정 단위로 분할하여 이를 개념 계층으로 추상화하였기 때문에 일반화된 데이터를 실세계로 사상하여 의미를 얻는 것은 불가능하다. 또한, MP[3,4]와 MPMine[11,12]에서도 R*-Tree를 이용하여 이동 객체의 위치값을 공간 개념 계층으로 일반화하는 방법을 제시하였다. 하지만 이러한 기법들이 제시한 일반화 방법은 공간 영역의 리스트인 시퀀스 형태로 객체의 이동 궤적을 표현함으로써 시퀀스를 구성하는 각 영역 내에서의 이동 이력이 모호하여 부정확한 이동 궤적을 제공할 수 있다. 가령, 공간상에서의 최적 경로 탐색이나 이동 객체의 순회 지점들에 대한 스케줄링 경로 예측과 같은 문제의 경우 가장 최적의 이동 패턴을 탐색하여 추출해야 하기 때문에 최하위 레벨의 공간 개념 계층에 속한 영역일지라도 영역 내의 패턴에 대한 분석이 필요하다.

본 연구는 이동 객체의 위치 이력 데이터 집합에서 나타날 수 있는 다양한 이동 패턴들 중 최적 이동 경로나 스케줄링 경로 예측을 위한 새로운 이동 패턴 탐사 기법을 개발하기 위한 것으로, 본 논문에서는 선행연구로써 MP[3,4]나 MPMine[11,12]에서 제시한 방법을 변형하여 이동 객체의 위치값을 공간영역에 대한 개념 계층으로 일반화하는 방법을 제안한다. 제안된 방법에서는 이동 객체의 연속적인 위치 변화를 보다 효과적으로 패턴화하기 위해서 레벨별 공간 영역과 이동 경로의 구간 분기점인 기점노드로 구성된 공간 개념 계층을 제시하고, R*-Tree와 AGHT를 사용하여 이동 객체의 상세 수준의 이력 데이터들을 공간영역으로 일반화한다. 이를 통해 객체의 이력 데이터 집합에 대한 패턴 탐사 수행 시간 및 탐사에 필요한 메모리 공간을 최소화함으로써 효율적인 패턴 탐사를 유도할 수 있다.

III. 이동 객체와 공간 개념 계층

3.1 이동 객체

이동 객체는 시간의 흐름에 따라 객체가 이동하면서 위치 및 모양이 연속적으로 변경되는 특성을 가지는 시공간 데이터이다. 이동 객체 데이터는 유일하게 식별되는 객체 식별자(*oid*)와 이산적으로 샘플링된 연속적인 이동 위치 데이터로

구성된다. 이동 위치 데이터는 2차원 공간상에서 샘플링된 객체의 위치(v)와 시간(t)으로 구성되며, 이동 객체의 위치 정보는 (oid, v, t)로 표현된다. v는 객체의 위치에 해당하는 (x, y) 좌표값이고, t는 유효시간을 나타내는 시간값이다 [14]. 본 논문에서는 다음과 같이 이동 객체를 정의한다.

[정의 1] 이동 객체 $MO=\langle OID, S, T, A\rangle$로 표현한다. OID는 객체 식별자, S는 공간속성, T는 시간속성, A는 객체의 속도, 방향, 상태 등 일반속성을 의미한다.

[정의 2] 이동 객체의 시간 속성은 유효 시간의 간격으로 구성되며, $T=\langle vt_{start}, vt_{end}\rangle$라 정의한다. 여기서, vt_{start}는 객체의 이동 시작 시각, vt_{end}는 객체의 이동 종료 시각이고, vt_{start}와 vt_{end}는 유효시간이다.

[정의 3] 이동 객체의 공간 속성은 특정 시간 속성 T 시점에서의 위치 좌표값을 나타내는 것으로, 시간 속성 $T=\langle vt_{start}, vt_{end}\rangle$ 일 때, 공간 속성 $S=\langle(x_{start}, y_{start}),(x_{end}, y_{end})\rangle$로 정의한다.

[정의 4] 이동 객체의 인스턴스 MO_i는 다음과 같이 정의한다. 이때, S_{i_k}는 MO_i의 k번째 공간 속성, T_{i_k}는 k번째 시간 속성, A_{i_k}는 k번째 일반 속성을 나타낸다.

$$MO_i=\langle OID_i, \{\langle S_{i_k}, T_{i_k}, A_{i_k}\rangle\}_{k=0}^{m}\rangle, \quad 0\le k\le m$$

3.2 공간 개념 계층

이동 객체의 위치 속성을 공간상의 특정 지역 수준의 의미로 일반화하기 위해서는 공간 영역에 대한 추상적인 공간 개념 계층이 필요하다. 공간 개념 계층은 실세계의 전체 공간을 상위 개념 수준에서 하위 개념 수준으로 특정 기준에 따라 각각 제한된 범위의 영역들로 구분한 추상 공간 계층이다. 즉 실세계의 전체지역을 분류함에 있어 지형이나 행정구획, 영역별 전용 유형 등 각기 차별화된 기준의 특성에 따라 각각의 영역을 여러 수준으로 구분한 형태이다. 본 논문에서는 공간 개념 계층을 행정구획에 따라 분류하고 최하위의 단말노드는 도로의 구간을 구분하는 기점노드로 구성한다.

[정의 5] 이동 객체가 이동 가능한 지역(Region)들의 집합 $R=\{R_i | R_i\in S,\ for\ ^{\forall} i\}$이며, 지역 R의 인스턴스 객체 $R_i=\langle R_{id}, R_S, R_A\rangle$로 표현한다. R_{id}는 지역 식별자이며, R_S와 R_A는 지역의 공간속성과 일반속성이다.

[정의 6] 공간 개념 계층 레벨의 집합 $L=\{L_j\}$이고, L_k 〈 L_{k-1}이다. 이 때, $0\le k\le j$ 이다.

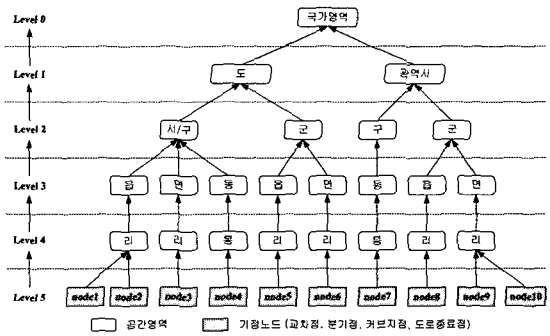
[정의 7] 공간 개념 계층 레벨 L에 대한 R의 인스턴스 객체는 다음과 같다.

$$L_{R_i}=\{L_{k_R} | L_{k_R}\subset R,\ L_{k_R}\in L_{k-1_R},\ 1\le k\le j \text{ and for } ^{\forall} i\}$$

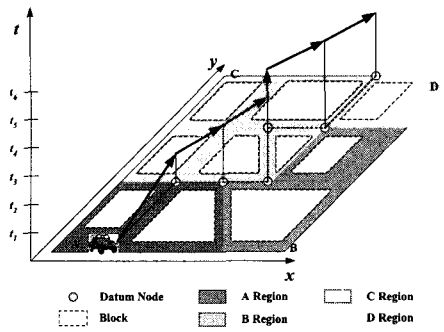
이러한 정의들에 의거하여 공간 지역에 대한 개념 계층을 생성할 수 있다. 다음 그림 1-(a)는 행정구획에 대한 6 레벨의 공간 개념 계층의 예로, 단말 수준의 노드는 도로 상의 교차점, 분기점, 커브지점, 도로종료점 등을 표현하고 각 레벨별 공간 범위는 포함관계를 통해 상위 레벨 영역으로 일반화된다. 그림 1-(b)는 시간 t축상에서 불규칙적인 단위 시간동안 단말노드인 기점노드 간을 이동하는 객체의 이력에 대한 표현 예이다.

IV. 이동 객체의 위치 일반화

위치 일반화는 2차원 공간 객체들 사이에 가능한 위상 관계를 분석하여 이동 객체의 위치를 공간영역으로 일반화한다. 공간 객체들 사이에서 나타날 수 있는 위상 관계들로는 OGC(Open GIS Consortium)의 구현 명세에서 정의한 Disjoint, Touches, Crosses, Within, Overlaps의 위상 관계 연산과 Contains, Intersects 라는 부가적인 위상 관계 연산이 있다. 이러한 위상 관계 연산들 중 이동 객체의 위치에 대한 공간 영역으로의 일반화를 위해서는 Contains 연산 정의를 이용할 수 있다. 즉, Contains 공간 연산을 통해 객체의 (x,y) 좌표점이 특정 영역에 포함되어 있는지를 검사하여 일정한 범위의 공간 영역으로 변환한다.



(a) 행정구역에 대한 개념 계층



(b) 공간영역과 기점노드
그림 1. 공간 개념 계층
Fig.1. Spatial Conception Levels

4.1 Contains 연산

이동 객체의 위치 일반화는 객체의 위치 좌표점이 공간 개념 계층에서 레벨 수준별 영역의 최소경계사각형(MBR: Minimum Bounding Rectangle)에 포함되었는지를 검사하는 단계와 특정 영역들의 경계에 위치하거나 또는 영역 내부에 포함되었는지를 검사하여 이동 객체의 위치 속성을 영역 범위로 일반화하는 단계를 거친다. 이 두 과정은 이동 객체의 위치 이력을 공간 영역으로 표현함에 있어 이동 객체가 샘플링되는 위치 좌표점을 영역 범위로 변환하는 Contains 공간 연산을 통해 처리된다. Contains 연산을 위한 선행 작업으로는 시공간 질의를 분석하여 공간 개념 계층의 레벨 수준을 결정해야 하는 과정이 필요하다.

```
Number of Spatial Concept Level: N
Set of Spatial Concept Level: Level = {L1, L2, ···, LN}
Number of Area MBR in Level Li: M
Set of Area MBR in Level Li : Li={MBR1, MBR2, ···, MBRm},   1≤i≤N
Bounding Point Set of Area MBR :
                    MBRj={BP(xmin,ymin), BP(xmax,ymin), BP(xmin,ymax), BP(xmax,ymax)},   1≤j≤M

Procedure Contains(MP,PreA,N)
Begin
        L={MBR1,MBR2,···,MBRN};
        CntMBR=∅;                          //Set of MBRs Contained MP

        For (i=1 to N)
                If (CntMBR ≠∅) Then
                        For(j=1 to CntMBRCount) Then
                                L=CntMBRj;
                                Low,CA=ContainedMBR(MP,L);
                        End For

                        CntMBR=Low,CntMBR;
                Else
                        CntMBR=ContainedMBR(MP,L);
                End If
        End For
        Return ContainedGrid(CntMBR,MP,PreA);
End
```

그림 2. Contains 알고리즘
Fig.2. Contains Algorithm

그림 2는 이동 객체의 위치값을 특정한 공간 영역 범위로 일반화하는 Contains 공간 연산 알고리즘으로, Contains 연산은 시공간 질의 분석을 통해 결정된 공간 계층 레벨 수준(N)까지 공간 영역을 반복적으로 MBR로 분할하며 ContainedMBR 함수를 통해 이동 객체의 위치 좌표점(MP)이 N 레벨 수준별 영역 MBR에 포함되는지를 검사한다. 이 때, 최상위 레벨 수준에서 하위 레벨 수준까지 MBR로 분할하는 이유는 시공간 질의 수준에 맞게 공간 개념 계층의 레벨 수준으로 이동 시퀀스를 생성하기 위함이다. 또한, MP가 포함된 모든 MBR들을 추출한 후 ContainedGrid 함수를 이용하여 MP가 실제 어느 영역의 경계 또는 내부에 위치하는지를 검사한다. 이 때, ContainedGrid 함수의 결과로 반환되는 MP를 포함한 영역은 이동 시퀀스를 생성하기 위해 일반화된 영역이다.

4.2 ContainedMBR 연산

위치 일반화를 위한 첫 단계는 공간 개념 계층의 레벨 수준에 따라 전체의 공간 영역을 레벨 수준별 영역의 MBR로 분할하여 각 MBR에 MP가 포함되었는지를 검사하는 단계이다. 그림 3과 같이 이동 객체가 A지점에서 B지점으로 이동 시 최적의 이동 경로를 탐색하려할 경우, 질의분석을 통해 결정된 레벨 수준 N=3 이라면 레벨 0 수준에서부터 레벨 N 수준까지 분할하여 N 레벨의 영역들로 이동 시퀀스를 생성해야 한다. 따라서 레벨 0 수준의 영역을 레벨 1 수준의 영역 L_1MBR로 분할한 후 각 영역에 대한 L_2MBR을 생성하고 A와 B를 포함하는 L_2MBR들을 추출한다. 그림 3에서 A는 L_2MBR_1, L_2MBR_2, L_2MBR_3에, B는 L_2MBR_4, L_2MBR_5, L_2MBR_6에 각각 포함된다.

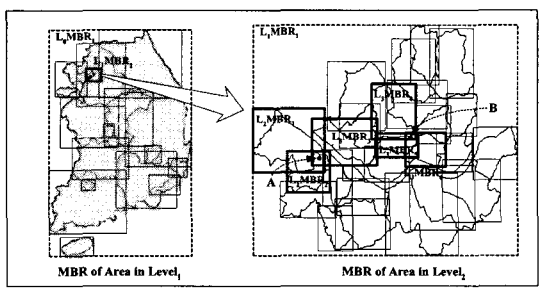


그림 3. L_1의 MBR에 대한 L_2의 MBR 표현
Fig.3. MBR Expression of the L_2 against MBR of the L_1

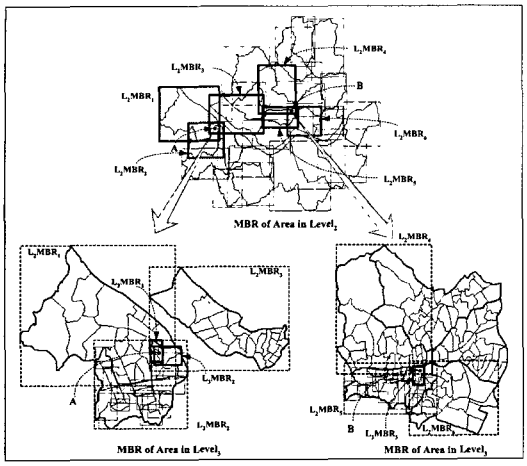


그림 4. L_2의 MBR에 대한 L_3의 MBR 표현
Fig.4. MBR Expression of the L_3 against MBR of the L_2

그림 4에서는 위치 좌표점이 포함되어 있는 레벨 2의 MBR에 대해 반복적으로 레벨 3의 MBR을 생성하여 위치 좌표점 A, B를 포함한 레벨 3 수준의 영역 MBR을 탐색한다. 좌표점 A와 B는 각각 L_3MBR_1, L_3MBR_2와 L_3MBR_3에 포함된다.

```
Procedure ContainedMBR(MP,Li)
Begin
        For(j=1 to Li.Count)
            If(MP contains Li.MBRj ||MP touchs Li.MBRj) Then
                ContMBR[ContMBRCount] = Li.MBRj;
            End If
        End For

        Return ContMBR;
End
```

그림 5. ContainedMBR 알고리즘
Fig.5. ContainedMBR Algorithm

그림 5의 ContainedMBR 연산은 이동 객체의 위치 좌표점(MP)과 각 레벨 수준별 영역 MBR 집합(L_i)을 입력받아 위치 좌표점이 각 MBR에 포함되었는지를 검사하여 해당 MBR들의 집합(ContMBR)을 반환하는 공간 연산 함수이다.

4.3 ContainedGrid 연산

이동 객체의 위치 일반화에서 두 번째 단계는 첫 번째 단계를 거쳐 추출한 레벨 수준별 영역 MBR들의 실제 영역에 대해서 이동 객체의 위치 좌표점이 영역의 경계에 위치하는지 또는 영역 내부에 포함되는지를 검사하여 이동 객체의 위치값을 영역 범위로 일반화하는 단계이다. 질의 분석을 통해 결정된 레벨 수준에 맞게 ContainedMBR 연산을 통해 MBR들을 추출한 후 그림 6과 같이 각 MBR을 특정 크기의 그리드(Grid)로 분할한다. MBR을 그리드로 분할하게 되면, MBR의 실제 영역에 완전 포함되는 셀과 영역 경계를 포함하는 셀, 실제 영역의 바깥 셀로 분류할 수 있다.

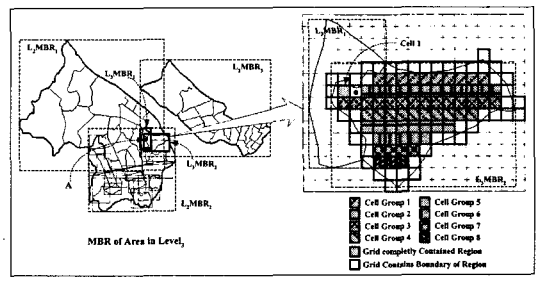


그림 6. L3의 MBR에 대한 Grid 분할
Fig.6. The Grid Division against MBR of the L_3

그림 6에서 영역을 그리드로 분할하는 이유는 각 분할된 그리드 셀(Cell)과 MP의 포함 여부를 검사하는 것이 전체 MBR 실영역 경계에 대해 검사하는 것보다 훨씬 효율적이기 때문이다. 즉, 영역에 완전 포함되는 셀의 경우 셀을 구성하는 세그먼트의 좌표 범위와 MP의 좌표점을 비교하면 되고, 실영역 경계를 포함하는 셀의 경우 셀의 사각 경계와 실제 영역 경계 세그먼트만을 비교하면 됨으로 연산시간을 감소시킬 수 있다.

그림 8의 ContainedGrid 연산에서는 MP가 각 셀에 포함되는지를 검사하여 완전 포함 셀이면 해당 MBR의 영역 이름을 반환하고, 영역 경계 포함 셀이면 ContainedArea 연산을 호출하여 셀 포함 여부를 검사하게 된다. 이 때, 각 셀에 대한 MP의 포함 여부를 검사하기 위해 그림 7과 같은 R*-Tree와 AGHT를 이용해 ContainedGrid 연산을 수행한다.

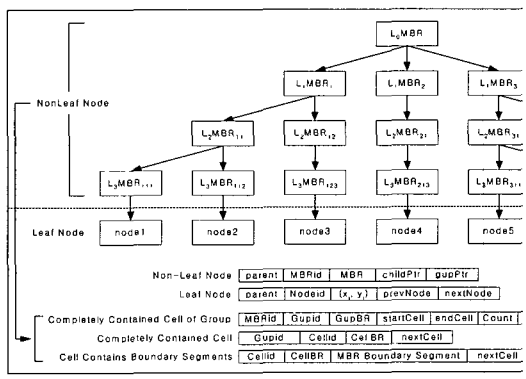


그림 7. R*-Tree + 영역 Grid 해쉬 테이블
Fig.7. R*-Tree + Area Grid Hash Table

그림 7에서 레벨별 MBR 영역을 의미하는 branch node는 해당 영역에 완전 포함되는 셀과 셀 그룹, 영역 경계 포함 셀의 정보를 관리하는 해쉬 테이블을 참조하기 위해 각각의 포인터를 갖는다. 서로 인접한 완전 포함 셀들은 셀 그룹으로 그룹화되며, 각 그룹은 그룹식별자(Gupid), 그룹의 경계 사각형(GupBR), 인접범위의 식별을 위한 최초 셀(startCell)과 마지막 셀(endCell), 최대인접 셀의 개수(Count), 다음 셀 그룹 및 셀에 대한 참조(nextGup, nextCell) 필드들로 구성된다. <그림 6>에서 L_3MBR_2는 8개의 셀 그룹과 1개의 완전 포함 셀로 구성된다. 이 때, 셀의 그룹화는 가능한한 MP와의 포함관계 연산 시 최소의 비교 횟수를 갖도록 그룹화한다. 또한, 각각의 완전 포함 셀은 셀이 포함된 그룹(Gupid), 셀 식별자(Cellid), 셀의 경계 사각형(CellBR), 다음 셀에 대한 참조(nextCell) 필드로 구성되며, 경계 포함 셀은 완전 포함 셀과 비슷한 필드로 구성되나 해당 셀을 참조하는 MBR의 경계 세그먼트 정보 필드를 추가로 갖는다.

```
Number of Area MBR in Level L : M
Number of Area Cell in MBR : K
Set of Area Cell in MBR : MBR={CellBR1, CellBR2, ···, CellBRK}, 1≤i≤M
Set of Area Cell Group in MBR : MBR={[CellGup]1,m, [cntCell]1,m, [lnkCell]1,m}
Set of Cell in Cell Gup : CellGup={cntCell}, 0≤a≤k
cntCell and lnkCell in MBR : cntCell=CellBRi, n=k,    lnkCelli=CellBRi, p=k
Binding Ring Set of Area CellBR :
                    CellBRi={CPi,1, ..., CPi,a, ..., CPi,n}, 1≤i≤M, 1≤a≤n

Procedure ContainedGrid(CntMBR,MPPrevA)
Begin
  String Answer;
  For(i=1 to CntMBRCnt)
    If ((CntMBR.CellGupCnt ≠0) || (CntMBRcompleteCellCnt=0)) Then
      For (j=1 to CntMBR.CellGupCnt)
        If ((MP exists CntMBR.CellGup) || (MP inCnt CntMBR.CellGup)) Then
          Answer = CntMBRno;
        End If
      End For
      For (k=1 to CntMBRcntCellCnt)
        If ((MP exists CntMBRcntCell) || (MP≤CntMBRcntCell)) Then
          Answer=CntMBRno;
        End If
      End For
    Else
      For (j=0 to CntMBRlnkCellCnt)
        If ((MP exists CntMBRlnkCell)|| (MP inCnt CntMBRlnkCell)) Then
          Answer=ContainedArea(CntMBRlnkCell,MPPrevA);
        End If
      End For
    End If
  End For
  Return Answer;
End
```

그림 8. ContainedGrid 알고리즘
Fig.8. ContainedGrid Algorithm

그림 8의 ContainedGrid 연산은 MP를 포함하는 MBR 집합(ContMBR), 이동 객체의 위치 좌표점(MP), 이동 객체의 이전 위치 영역(PrevA)에 대한 정보를 전달받아 각 영역 MBR에 대응하는 실영역들이 위치 좌표점을 포함하는지 검사하여 해당 영역의 이름을 반환하는 공간 연산 함수이다. ContainedGrid 연산에서는 먼저, 각 영역 MBR에 대한 완전 포함 셀 그룹과 완전 포함 셀에 MP가 완전 포함되거나 또는 셀 경계 상에 존재하는지를 검사하여 조건에 만족한다면 해당 영역의 이름을 반환한다. 만약, MP가 두 조건을 만족하지 못한다면, MBR의 경계 세그먼트를 포함하는 셀들을 탐색한다. 조건을 만족하는 셀이 존재한다면 ContainedArea 연산에 해당 셀을 전달하여 실제 경계 포함 셀에 MP가 포함되는지를 검사한다.

4.4 ContainedArea 연산

ContainedGrid 연산으로부터 ContainedArea 연산에 전달되는 셀의 정보 중 셀의 경계는 완전 포함 셀과 같이 셀의 경계 사각형(CellBR)이 아닌 셀의 경계 사각형과 MBR 세그먼트 객체의 조합으로 구성된 새로운 셀 경계이다. 즉, 그림 9에서 MP가 존재하는 경계 포함 셀 C(0,0), C(0,1), C(1,4), C(2,1), C(2,4), C(3, 3)을 보면 두꺼운 실선으로 표현된 부분이 실제 셀 영역의 경계가 된다. 이러한 셀의 실영역 내에 MP가 포함되는지에 대한 검사는 MP로부터 시작하는 단방향 직선 l을 이용하는 방법으로, 직선 l과 셀 영역 간선(경계선)들 간의 교차 여부를 통해 영역 내에 점이 포함되는지를 검사한다. 직선 l과 교차하는 영역 간선들의 수가 홀수 개이면 영역에 포함되고 0 이거나 짝수 개이면 영역 외부에 존재하는 것으로 판단한다. 단, 두 개 이상의 간선이 만나는 한 끝점을 직선 l이 통과할 경우, 각 간선의 또 다른 끝점이 l보다 높게 위치하면 교차하는 간선으로, 낮게 위치하면 교차하지 않는 간선으로 간주하며, 직선 l이 임의의 한 간선과 동일선 상에 위치할 경우에도 교차하지 않는다고 간주한다.

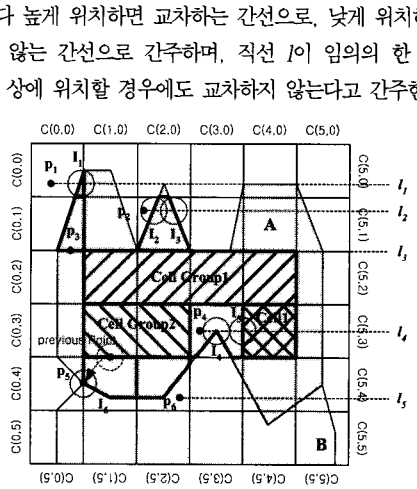


그림 9. 공간 영역 A의 셀과 단방향 직선 l과의 교차에 따른 포함 관계
Fig.9. Inclusion Relationship on the Intersection Cells of Area A with Line *l*

그림 9에서 P_1에서부터 시작하는 직선 l_1은 셀 C(0,0)의 영역 간선들과 I_1에서 교차하고 P_2에서 시작하는 직선 l_2는

셀 C(2,1)의 영역 간선들과 I_2, I_3에서 교차하는데, I_1, I_2, I_3 모두 교차하는 간선으로 간주하여 l_1과 l_2는 교차하는 간선의 수가 각각 총 2개이므로 셀 C(0,0)과 C(2,1)의 영역 밖에 위치한다. P_3에서 시작하는 직선 l_3의 경우 셀 C(0,1)의 한 영역 간선 상에 위치함으로 C(0,1)의 영역에 포함된다. 또한, P_4에서 시작하는 직선 l_4는 I_4, I_5에서 교차하게 되는데, I_4의 경우 두 간선이 만나는 한 끝점을 직선 l_4가 통과하지만 두 간선의 양 끝점 모두 직선 l_4보다 높게 위치하지 않으므로 교차하지 않는 간선으로 간주하고, I_5의 경우만을 교차하는 간선으로 간주한다. 마지막 P_5는 두 개 이상의 영역들이 만나는 간선 상에 MP가 위치하는지를 검사하는 예로, 만약 영역 간선 상에 존재한다면 이동 객체가 이전에 위치했던 영역이 어디인지 판단하여 이전 영역에 포함된다고 간주한다. 즉, 그림 9에서 P_5는 영역 A와 영역 B 사이의 간선 상에 위치함으로 이전에 이동 객체가 위치한 영역인 A에 포함된다.

```
Cell Area contained CellBR : A = {E1,E2,E3,···,En}.  A={AR1,AR2,AR3,···,ARn}
End Points of Area Edge i : E={AP1,APn}.  En={APn, AR1},  1 ≤ i <n
Horizontal pass across the MP : l

String ContainedArea(bounCell, MP, PrevA)
Begin
        String Areaname;
        Int count = 0;

        A = isAreaEdge(bounCell);
        For(j=1 to A.Count)
            If(onEdge(A.Ej, MP) == true) Then
                If(PrevA.Name ≠ Null)
                    Areaname = PrevA.Name;      Break;
                Else
                    Areaname = A.Name;          Break;
                End If
            Else
                If(Intersect(A.Ej,l) == true)
                    count = count + 1;
                End If
            End If
        End For

        If(count ≠ 0) And (count is odd) Then
            Anamename = A.Name;
        End IF

        Return Areaname;
End
```

그림 10. ContainedArea 알고리즘
Fig.10. ContainedArea Algorithm

그림 10의 ContainedArea 연산은 MBR에 대한 경계 포함 셀(bounCell)과 이동 객체의 위치 좌표점(MP), 이동 객체의 이전 위치 포함 영역(PrevA)을 전달받아 MBR 경계를 포함하는 그리드의 셀 영역에 객체의 위치 좌표점이 포함되는지를 검사하는 공간 연산 함수이다. 연산의 처리 과정은 먼저, isAreaEdge 함수를 통해 셀의 경계 사각형(CellBR)과 MBR의 경계 세그먼트(Boundary Segment)로 조합된 새로운 셀의 실 영역에 대한 간선을 구성하고, 모든 셀의 실영역 간선 정보들을 추출한다. 셀의 실영역에 대한 간선 정보가 추출되면, onEdge 함수는 각 간선 상에 MP가 위치하는지를 검사한다. 만약 MP가 간선 상에 존재한다면 이전 MP가 위치했던 영역의 이름을 반환하고, 존재하지 않는다면 Intersect 함수에 의해 교차 간선의 수를 계산한다. Intersect 함수는 MP에서부터 시작하는 수평 단방향 직선 l이 셀의 영역 간선들과 교차하는지에 대한 여부를 판별하는 공간연산 함수이다. ContainedArea 연산의 마지막 과정은 Intersect 함수에 의해 판별된 교차 간선의 총 개수가 0 또는 짝수인지 혹은 홀수인지를 계산하여 홀수일 경우 MP가 해당 셀 영역에 포함된다고 간주하여 영역의 이름을 반환한다.

4.5 Intersect 연산

다음 그림 11의 Intersect 연산은 경계 포함 셀의 실 영역을 구성하는 하나의 간선 E와 직선 l의 값을 입력받아 두 선이 교차하는지 또는 직선 l과 간선 E가 동일선 상에 위치하는지를 검사하는 공간 연산 함수이다. 이 때 수평 단방향 직선 l은 MP에서부터 셀의 경계 사각형(CellBR)의 우측 경계선을 구성하는 한 점까지의 직선으로 $l = \{MP, CBP_n \in \{above\_right, below\_right\}\}$ 로 표현된다.

```
End Points of Area Edge i : Ei={APi, APn}.  En={APn, AR1}.  1≤i<n
Set of Points of Area Edge i : Ei={P1,P2,P3,···,Pn}.  1≤j≤n
                                    If (j<n) Then  P1=APi and Pn=APn ,
                                    If (j=n) Then  P1=APi and Pn=AR1
Point of Area Edge i : P=(xn, yn),  1≤k≤m
BoundingPoint Set of Area CellBR :
                    CellBR1={CBP1above_right, CBP1below_right, CBP1below_left, CBP1above_left},  1≤j≤M
Horizontal pass across the MP: l={MP, CBPn∈{above_right, below_right}}
Point of Moving Object : MP=(x, y)
Any Point of Right Bounding Line:
                                    CBPn∈{above_right, below_right} =(xi∈{above_left, below_left}, y)

Boolean Intersect(Ei, l)
Begin
        Boolean b=false;
        Long x_coord;
        Long y_coord=MPy;

        If ((onEdge(l, P1)∧onEdge(l, P2)) ≠true) Then
            For (x_coord=MPx to BPn∈{above_right, below_right}.x)
                For(i=1 to n)
                    If ((x_coord==Pi.x)∧(y_coord==Pi.y)) Then
                        b= true;
                    End If
                End For
            End For
        Else If ((onEdge(l, P1)==true) ‖(onEdge(l, P2)==true)) Then
            If ((P1.y>y_coord) ‖(P2.y>y_coord)) Then
                b=true;
            End If
        End If

        Return b;
End
```

그림 11. Intersect 알고리즘
Fig.11. Intersect Algorithm

Intersect 연산 과정에서 onEdge 함수는 직선 l과 점들의 집합 $\{P_1, P_2, P_3, \cdots, P_n\}$으로 표현될 수 있는 셀의 영역 간선 E의 한 점 P를 입력받아 직선 l 상에 점 P가 위치하는지 검사한다. 만약 간선 E의 시작점 P_1과 끝점 P_n이 직선 l 상에 모두 존재하지 않는다면 두 선을 구성하는 각각의 점이 서로 같은 지점에 위치하는지를 반복적으로 검사하여 같을 경우 True 값을 반환한다. 반대로 간선 E의 두 점 P_1과 P_n이 직선 l 상에 모두 존재한다면 직선 l과 간선 E는 동일선 상에 존재함으로 교차되지 않는다고 간주되어 False 값을 반환한다. 또한, 간선 E의 두 점 P_1과 P_n 중 하나의 점만이 직선 l 상에 존재할 경우 다른 점이 l과의 교차점보다 위에 있으면 True 값을 아래에 있으면 False값을 반환한다.

V. 실험 및 성능평가

5.1 알고리즘 구현 및 실험

위치 일반화 과정을 수행하기 위해서 이동 객체 데이터베이스는 객체 식별자를 주키로, 트랜잭션 시간을 보조키로 정렬한다. 실험을 위한 이동 객체의 위치 정보 획득은 공간 개념 계층의 최하위 레벨인 기점노드에서 샘플링하였다고 가정한다. 다음 그림 12와 같이 공간 지역과 도로망이 있다고 가정한다.

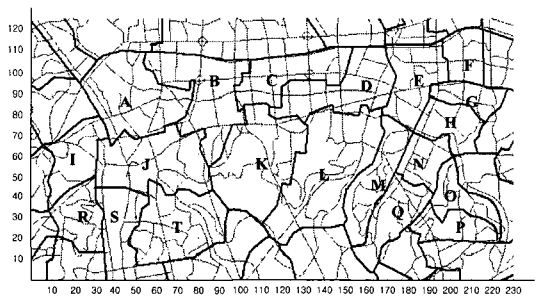


그림 12. 공간지역과 도로망 예
Fig.12. Example of Spatial Region and Road Network

다음 표 1-(a)는 객체의 ID별로 객체의 공간값 속성을 샘플링된 시간순으로 정렬한 데이터베이스이고, 표 1-(b)는 각 객체의 공간값 속성이 어떠한 영역에 포함되는지 Contains 공간 연산을 적용하여 일반화된 영역으로 변환한 예이다. 표 1-(b)와 같이 이동 객체의 위치 속성이 공간영역으로 일반화 되면 이동 객체의 이동 변화를 시간에 따라 순차적으로 나열하여 이동 시퀀스를 생성할 수 있다. 이동 시퀀스의 생성은 시간 순차의 지속기간(duration) 및 특정 공간 범위로 제한된 영역의 순차리스트로 표현할 수 있으나, 이러한 영역의 순차리스트는 사건(영역 변화) 사이의 시간 간격이 고려되지 않아 패턴 마이닝을 위한 트랜잭션인 이동 시퀀스로 사용하기에는 문제가 있다. 일반적으로 패턴 탐사의 대상이 되는 하나의 시퀀스는 패턴에 있는 사건들 사이의 시간 간격(interval)을 만족해야만 하나의 트랜잭션, 즉 시퀀스로 생성될 수 있다. 따라서 이동 패턴 마이닝의 대상이 되는 이동 시퀀스를 생성하기 위해서는 시퀀스를 구성하는 영역 간에 최대 시간 간격에 대한 제약조건을 두어 이를 만족해야만 의미있는 이동 시퀀스로 생성될 수 있도록 해야 한다. 즉, 이동 객체의 공간 속성에 대한 샘플링 시간을 검사하여 특정 영역에 머문 시간이 최대 시간 간격 max_gap을 초과하면 초과 이전까지의 이동 시퀀스와 초과 이후의 이동 시퀀스로 분리한다.

표 1. Contains 연산을 통한 일반화 영역으로의 변환
Table 1. Generalized Area converting by Contains Operation

(a) 정렬된 데이터베이스

OID	VT	X	Y
1	2006/05/08/11/05	62	91
	2006/05/08/11/22	87	87
	2006/05/08/11/35	117	74
	2006/05/08/11/53	158	66
	2006/05/08/12/01	171	55
	2006/05/08/12/13	186	40
	2006/05/08/12/36	211	38
2	2006/05/09/07/45	29	76
	2006/05/09/08/35	71	66
	2006/05/09/08/48	89	74
	2006/05/09/08/59	100	70
	2006/05/11/06/51	100	70
	2006/05/11/07/24	126	29
	2006/05/11/08/02	156	46
	2006/05/11/08/38	191	47
3	2006/05/07/09/04	33	25
	2006/05/07/09/43	50	28
	2006/05/07/10/05	75	36
	2006/05/07/10/55	91	51
	2006/05/07/11/26	120	36
	2006/05/07/12/11	153	33
	2006/05/07/12/48	163	18
4	2006/05/09/10/20	48	92
	2006/05/09/10/48	69	95
	2006/05/09/11/05	116	89
	2006/05/09/11/26	148	90
	2006/05/09/11/58	172	89
	2006/05/09/12/21	207	94
	2006/05/09/12/46	207	105
5	2006/05/10/17/15	75	49
	2006/05/10/18/28	118	45
	2006/05/10/18/53	134	79
	2006/05/10/19/36	148	83
	2006/05/10/19/59	175	88
	2006/05/10/20/19	190	79
	2006/05/10/20/38	194	91
6	2006/05/09/20/10	44	28
	2006/05/09/20/36	59	59
	2006/05/09/20/49	89	74
	2006/05/09/21/03	105	91
	2006/05/09/21/27	123	71
	2006/05/09/21/39	134	79
	2006/05/09/22/05	148	83

(b) 위치 일반화 예

OID	VT	Area
1	2006/05/08/11/05	A
	2006/05/08/11/22	B
	2006/05/08/11/35	K
	2006/05/08/11/53	L
	2006/05/08/12/01	M
	2006/05/08/12/13	Q
	2006/05/08/12/36	O
2	2006/05/09/07/45	I
	2006/05/09/08/35	J
	2006/05/09/08/48	B
	2006/05/09/08/59	K
	2006/05/11/06/51	K
	2006/05/11/07/24	L
	2006/05/11/08/02	M
	2006/05/11/08/38	N
3	2006/05/07/09/04	R
	2006/05/07/09/43	S
	2006/05/07/10/05	T
	2006/05/07/10/55	K
	2006/05/07/11/26	L
	2006/05/07/12/11	M
	2006/05/07/12/48	Q
4	2006/05/09/10/20	A
	2006/05/09/10/48	B
	2006/05/09/11/05	C
	2006/05/09/11/26	D
	2006/05/09/11/58	E
	2006/05/09/12/21	G
	2006/05/09/12/46	F
5	2006/05/10/17/15	J
	2006/05/10/18/28	K
	2006/05/10/18/53	L
	2006/05/10/19/36	D
	2006/05/10/19/59	E
	2006/05/10/20/19	H
	2006/05/10/20/38	G
6	2006/05/09/20/10	S
	2006/05/09/20/36	J
	2006/05/09/20/49	B
	2006/05/09/21/03	C
	2006/05/09/21/27	K
	2006/05/09/21/39	L
	2006/05/09/22/05	D

7	2006/05/10/18/03	206	102
	2006/05/10/18/36	206	86
	2006/05/10/18/55	201	75
	2006/05/10/19/24	192	65
	2006/05/12/08/14	192	65
	2006/05/12/08/51	177	48
	2006/05/12/09/33	187	29
	2006/05/12/10/08	199	35
8	2006/05/11/18/03	97	97
	2006/05/11/18/36	132	99
	2006/05/11/18/55	163	88
	2006/05/11/19/24	184	82
	2006/05/12/08/14	184	82
	2006/05/12/08/51	191	82
	2006/05/12/09/33	182	60
	2006/05/12/10/08	171	40
9	2006/05/12/22/03	69	35
	2006/05/12/22/46	83	47
	2006/05/12/23/25	94	59
	2006/05/14/06/58	94	59
	2006/05/14/07/22	102	74
	2006/05/14/07/49	100	83
	2006/05/14/08/32	98	90
	2006/05/14/08/32	62	87
	2006/05/14/08/32	55	57

7	2006/05/10/18/03	F
	2006/05/10/18/36	G
	2006/05/10/18/55	H
	2006/05/10/19/24	N
	2006/05/12/08/14	N
	2006/05/12/08/51	Q
	2006/05/12/09/33	P
	2006/05/12/10/08	O
8	2006/05/11/18/03	B
	2006/05/11/18/36	C
	2006/05/11/18/55	D
	2006/05/11/19/24	E
	2006/05/11/08/14	E
	2006/05/12/08/51	H
	2006/05/12/09/33	N
	2006/05/12/10/08	Q
9	2006/05/12/22/03	T
	2006/05/12/22/46	J
	2006/05/12/23/25	K
	2006/05/14/06/58	K
	2006/05/14/07/22	B
	2006/05/14/07/49	C
	2006/05/14/08/32	B
	2006/05/14/08/32	A
	2006/05/14/08/32	J

가령, 표 1-(b)에서 객체 2의 이동 경로는 영역 간을 이동한 시간 간격을 고려하지 않았을 경우 I→J→B→K→L→M→N 으로 표현될 수 있다. 만약 영역 간의 최대 시간 간격 max_gap이 최대 1일이라면 영역 K에서 멈추었다가 다시 L를 향해 출발하는데 걸린 시간이 1일 이상이기 때문에 이동 경로 I→J→B→K→L→M→N은 〈I J B K〉와 〈K L M N〉, 두 개의 시퀀스로 분리되어야 한다. 마찬가지로 객체 7, 9의 이동 경로는 각각 F→G→H→N→Q→P→O, T→J→K→B→C→B→A→J로 표현되나 객체 7의 이동 시퀀스는 〈F G H N〉, 〈N Q P O〉로, 객체 9의 시퀀스는 〈T J K〉, 〈K B C B A J〉로 분리된다. 하지만 객체 8의 경우 객체가 E 지점에서 멈추었다 다시 H를 향해 출발하지만 E에서 H로의 출발시간이 최대 시간 간격 1일을 초과하지 않음으로 B→C→D→E→H→N→Q로 표현된다. 만약 객체 8의 시퀀스를 분리하기 위해 최대 시간 간격 max_gap을 6시간 정도로 제한한다면 〈B C D E〉와 〈E H N Q〉, 두 개의 시퀀스로 분리할 수 있다. 이와 같이 최대 시간 간격 max_gap에 대한 설정이 적절하지 않다면, 사건(영역 변화) 사이의 시간 간격이 제대로 고려되지 않아 단 하나의 영역 순차리스트로 표현되기 때문에 패턴 마이닝을 위한 이동 시퀀스로 사용하기에는 적합하지 못하다. 다음 표 2는 표 1-(b)의 일반화된 영역을 시간 간격을 고려하여 이동 시퀀스로 생성한 것이다.

표 2. 이동 시퀀스(트랜잭션 데이터)
Table 2. Moving Sequence

OID	Moving Sequence
1	〈A B K L M Q O〉
2	〈I J B K〉, 〈K L M N〉
3	〈R S T K L M Q〉
4	〈A B C D E G F〉
5	〈J K L D E H G〉
6	〈S J B C K L D〉
7	〈F G H N〉, 〈N Q P O〉
8	〈B C D E〉, 〈E H N Q〉
9	〈T J K〉, 〈K B C B A J〉

그림 13은 Contains 공간 연산 알고리즘의 구현을 통해 표 1-(a)의 이동 객체 위치 정보를 일반화하여 이동 시퀀스를 생성하는 프로그램의 실행 결과이다.

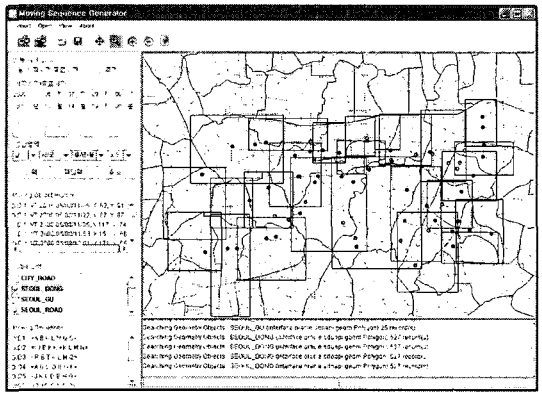


그림 13. 위치 일반화 실행 결과
Fig.13. Execution Results of Location Generalization

5.2 공간 연산 알고리즘의 성능평가

다음 표 3과 그림 14는 위의 실험 데이터를 이용하여 기존의 이동 패턴 탐사 기법 중 공간 영역에 대한 일반화를 위해서 MP[3,4]와 MPMine[11,12] 기법에서 제안한 공간 연산 알고리즘과 본 논문에서 제시한 알고리즘을 공간영역으로의 일반화 수행에 따른 소요시간을 기준으로 비교한 결과이다.

표 3. 공간영역으로의 일반화 수행에 따른 소요시간
Table 3. Execution Time in Response to Perform Generalization into Spatial Region

수행시간(초)	A	B	C	D	E	F	G	H	I	J
MP& MPMine	3.14	4.15	2.99	3.24	3.17	2.68	2.77	3.08	3.27	4.21
Contains	1.12	1.29	1.06	1.16	1.13	0.87	0.93	1.04	1.14	1.31
수행시간(초)	K	L	M	N	O	P	Q	R	S	T
MP& MPMine	4.53	4.78	3.77	2.81	3.64	3.69	3.87	3.69	3.78	3.99
Contains	1.51	1.55	1.19	1.01	1.17	1.18	1.21	1.18	1.22	1.27

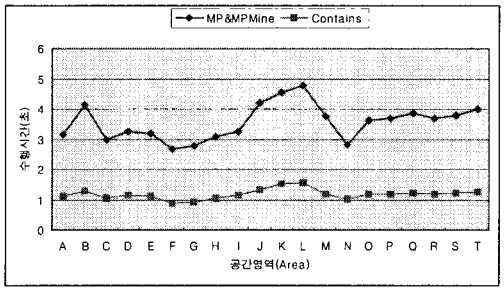


그림 14. 공간영역으로의 일반화 수행에 따른 성능비교
Fig.14. Performance Comparison to perform Generalization into Spatial Region

그림 14에서 MP&MPMine의 공간 연산 알고리즘의 경우 B, J, K, L 영역으로의 일반화 수행 시 소요시간이 많이 걸리는 것을 알 수 있다. 이는 각 공간 영역을 구성하는 경계 세그먼트의 수가 다른 영역에 비해 많은 경우로 MP&MPMine의 알고리즘은 각 세그먼트들에 대해서 모두 교차 여부를 검사하기 때문이다. 따라서 경계 세그먼트의 수가 N일 때 MP&MPMine의 알고리즘은 최대 $O(n)$의 시간 복잡도를 갖는다.

반면 본 논문에서 제안한 contains 알고리즘의 경우 비교적 일정한 수행시간을 보이는데, 이는 모든 공간영역에 대한 일정한 크기의 셀로 분할하기 때문에 분할된 각 셀과 셀 그룹에 대한 이동점 포함 여부 검사시간이 일정하기 때문이다. contains 알고리즘은 공간영역을 일정한 크기의 셀로 분할하여 셀 그룹과 셀의 포함여부를 먼저 검사하고 결정된 이동점을 포함하는 셀의 경계 세그먼트에 대해서만 교차 여부를 검사하기 때문에 MP&MPMine의 알고리즘보다 훨씬 적은 수행시간을 소요한다. 또한 실영역 경계 포함 셀의 경우 경계 세그먼트의 수가 실영역의 경계 세그먼트의 수보다 훨씬 적기 때문에 MP&MPMine의 알고리즘의 수행시간에 비해 비교적 효율적인 성능을 보인다.

VI. 결론 및 향후 연구과제

이동 객체의 위치 이력 데이터들은 공간상에서 x, y 좌표값으로 표현되기 때문에 낮은 수준의 상세한 정보 형태를 띠고 있다. 하위 개념 수준으로 표현된 위치 데이터들로부터 직접적으로 지식화 가능한 패턴들을 탐색하는 것은 매우 어려운 일이기 때문에 이동 객체의 연속적 위치 변화를 보다 효과적으로 패턴화하기 위한 공간영역으로의 일반화 접근법이 필요하다.

따라서 본 논문에서는 이동 객체의 위치값과 공간 영역 간의 위상 관계를 고려하여 레벨별 공간 영역과 이동 경로 구간의 분기점인 기점노드로 구성된 추상 공간 개념 계층을 정의하였고, 공간 인덱싱 구조인 R*-Tree와 이에 연결된 MBR 영역에 대한 AGHT를 기반으로 상세 수준의 위치 이력 데이터들을 의미있는 공간영역으로 일반화하는 방법을 제시하였다. MBR 영역에 대한 AGHT는 MBR 간의 겹침의 최소화와 영역의 최소화에 기반한 최적화 인덱스인 R*-Tree의 Branch Node에 의해 참조되는 것으로, 각 레벨별 공간 영역을 그리드로 분할한 후 각 영역에 완전 포함되는 셀과 셀 그룹, 실영역 경계 포함 셀로 구성하였다. 이러한 R*-Tree와 AGHT의 구조는 이동 객체의 위치값이 특정 영역에 포함되었는지를 검사하기 위한 기존의 MP[3,4]와 MPMine[11,12] 기법에서 영역 경계를 구성하는 폴리건의 모든 세그먼트와 비교 연산을 수행하는 방법에 비해 영역 완전 포함 셀과 실영역 경계 포함 셀에 대한 비교 연산만을 수행하도록 하여 연산 처리 시간을 크게 감소시켰다. Contains 공간 연산은 일반화된 데이터 집합을 생성하는데 있어 이를 이용함으로써 보다 효과적인 이동 패턴 탐사를 지원할 수 있다.

또한, Contains 연산은 이동 객체의 위치 정보가 공간 개념 계층에서 레벨 수준별 영역의 MBR에 포함되었는지를 검사하는 ContainedMBR 연산과 R*-Tree+ AGHT 구조를 이용하여 위치 정보가 특정 MBR을 구성하는 셀의 영역 내에 포함되었는지를 검사하는 ContainedGrid 및 ContainedArea 연산으로 구성하였다. 이러한 Contains 공간 연산을 통해 일반화된 각 공간 영역 데이터를 이용하여 이동 시퀀스를 생성하였으며, 이렇게 생성된 이동 시퀀스들은 의미있는 지식 추출을 위한 이동 패턴 탐사에 이용될 수 있고, 패턴 탐사를 통한 객체의 이동 추이 분석을 통해 다양한 형태의 서비스로 개발될 수 있다.

향후 연구과제로는 공간영역으로의 일반화 방법을 통해 생성된 일반화된 이동 객체 데이터와 이동 시퀀스를 이용하여 최적 이동 경로를 탐색하기 위한 패턴 탐사 기법의 개발이 요구된다. 또한 최적 이동 경로 탐색을 위한 패턴 탐사 기법을 이용하여 단위 시간동안 이동 객체가 순회해야 하는 지점들에 대한 스케줄링 경로 예측을 위한 마이닝 기법의 개발도 필요하다.

참고문헌

[1] N. Mamoulis, H. Cao, G. Kollios, M. Hadjieleftheriou, Y. Tao, and D. W. Cheung, "Mining, Indexing, and Querying Historical Spatio-Temporal Data", Proc. of the International Conference on Knowledge Discovery and Data Mining, 2004.

[2] P. Stolotz, and H. Nakamura, "Fast Spatio-Temporal Data Mining of Large Geophysical Datasets", Proc. of the International Conference on Knowledge Discovery and Data Mining, 1995.

[3] J. D. Chung, O. H. Paek, J. W. Lee , K. H. Ryu,

"Temporal Pattern Mining of Moving Objects for Location-Based Service", Proc. of the 13th International Conference on Database and Expert Systems Applications, September 02-06, 2002.

[4] J. W. Lee, O. H. Paek, K. H. Ryu, "Temporal moving pattern mining for location-based service", The Journal of Systems and Software, Vol.73. 2004.

[5] S. Ramaswamy, S. Mahajan and A. Silberschatz, "On the discovery of interesting patterns in association rules", the VLDB Conference, New York City, September 1998.

[6] H. Manniila, H. Toivonen, and A. I. Verkamo, "Discovery of frequent episodes in event sequences", Data Mining and Knowledge Discovery, 1(3), pp.259-289, November, 1997.

[7] R. Agrawal and R. Srikant, "Mining sequential patterns", In Proc. 11th International Conference on Data Engineering, 1995.

[8] R. Srikant and R. Agrawal, "Mining sequential patterns : Generalizations and Performance Improvements", International Conference on Extending Database Technoloty, Springer-verlag, 1996.

[9] E. Tsoukatos and D. Gunopoulos, "Efficient Mining of Spatio-Temporal Patterns", Proc. of the 7th International Symposium on Spatial and Temporal Database(SSTD), pp.425-442, 2001.

[10] J. Han, G. Dong, and Y. Yin, "Efficient Mining of Partial Periodic Patterns in Time Series Database", Proc. of The 11th International Conference on Data Engineering, 1999.

[11] 이준욱, 남광우, "이동 객체 위치 일반화를 이용한 시공간 이동 패턴 탐사", 한국정보처리학회 논문지, 제10-D권, pp.1103-1114, 제7호, 2003.

[12] 이준욱, "지식 탐사 프레임워크 기반의 시공간 이동 패턴 탐사 기법", 박사학위논문, 충북대학교 대학원, 2003.

[13] 박지웅, "시공간 이동 패턴 추출을 위한 효율적인 알고리즘", 박사학위논문, 건국대학교 대학원, 2006.

[14] J. Moreira, C. Ribeiro, and J. M. Saglio, "Representation and Manipulation of Moving Points : An Extended Data Model for Location Estimation", Cartography and Geographic Information System (CaGIS), ACSM, Vol.26, No.2, April 1999.

저 자 소 개

고 현 (Hyun Ko)

2001년 군산대학교 컴퓨터정보과학과 이학사

2003년 군산대학교 컴퓨터정보과학과 이학석사

2004년~현재 군산대학교 컴퓨터정보과학과 박사과정

※ 관심분야 : 에이전트, 시공간 데이터베이스, 시공간 데이터 마이닝, 이동객체 시스템

김광종 (Kwang-Jong Kim)

1993년 군산대학교 컴퓨터정보과학과 이학사

1999년 군산대학교 컴퓨터정보과학과 이학석사

2003년 군산대학교 컴퓨터정보과학과 이학박사

※ 관심분야 : 에이전트, 분산객체시스템, 데이터 마이닝 능동 데이터베이스

이연식 (Yon-Sik Lee)

1982년 전남대학교 전자계산학과 이학사

1984년 전남대학교 전자계산학과 이학석사

1994년 전북대학교 전산응용공학과 공학박사

1995년~1997년 군산대학교 교무부처장

1997년~1998년 University of Missouri 교환교수

1999년~2001년 군산대학교 전자계산소장

2004년~2005년 Ohio State University 교환교수

1998년~현재 군산대학교 컴퓨터정보과학과 교수

※ 관심분야 : 번역기 이론, 객체지향시스템, 능동시스템, 지능형 에이전트, 데이터 마이닝