시공간 이동 패턴 추출을 위한 효율적인 알고리즘†

An Efficient Algorithm for Spatio-Temporal Moving Pattern Extraction

박지웅 * / Ji-Woong Park 홍동숙 *** / Dong-Suk Hong
김동오 ** / Dong-Oh Kim 한기준 **** / Ki-Joon Han

요 약

최근 들어 이동 객체의 이력(history) 데이타에서 이동 객체의 이동 패턴, 즉 연속되는 시간 영역에서 반복적으로 발생되는 공간 이동 경로와 같은 다양한 지식을 추출하여 활용하는 응용 서비스의 활용성이 점점 증대되고 있다. 그러나 기존의 이동 패턴 추출 방법은 최소지지도(minimum support)가 낮은 경우에 많은 수의 후보 이동 패턴이 생성되고 이로 인하여 수행 시간과 소요 메모리가 급격히 증가하게 되는 단점이 있다.

본 논문에서는 대용량의 시공간 데이타 집합으로부터 이동 객체의 이동 패턴을 효율적으로 추출하기 위한 STMPE(Spatio-Temporal Moving Pattern Extracting) 알고리즘을 제안한다. STMPE 알고리즘은 시공간 데이타를 일반화시킴으로서 메모리 사용량을 최소화할 수 있으며, 단기 이동 패턴을 작성하여 유지하기 때문에 데이타베이스 스캔 횟수를 최소화할 수 있다. STMPE 알고리즘은 모든 부분에서 시간 정보를 갖는 다른 시공간 이동 패턴 추출 알고리즘보다 최소지지도가 낮아질수록, 이동 객체의 수가 증가할수록, 시간 분할 횟수가 많아질수록 더욱 뛰어난 성능을 보였다.

Abstract

With the recent the use of spatio-temporal data mining which can extract various knowledge such as movement patterns of moving objects in history data of moving object gets increasing. However, the existing movement pattern extraction methods create lots of candidate movement patterns when the minimum support is low.

Therefore, in this paper, we suggest the STMPE(Spatio-Temporal Movement Pattern Extraction) algorithm in order to efficiently extract movement patterns of moving objects from the large capacity of spatio-temporal data. The STMPE algorithm generalizes spatio-temporal and minimizes the use of memory. Because it produces and keeps short-term movement patterns, the frequency of database scan can be minimized. The STMPE algorithm shows more excellent performance than other movement pattern extraction algorithms with time information when the minimum support decreases, the number of

† 본 연구는 정보통신부 및 정보통신연구진흥원의 대학 IT 연구센터 육성·지원사업의 연구결과로 수행되었음.
■ 논문접수 : 2006.8.1 ■ 심사완료 : 2006.9.11

* 경기공업대학 컴퓨터정보시스템과 조교수(jiwpark@kinst.ac.kr)
** 건국대학교 컴퓨터공학과 강의교수(dokim@db.konkuk.ac.kr)
*** 건국대학교 컴퓨터공학과 박사과정(dshong@db.konkuk.ac.kr))
**** 교신저자 건국대학교 컴퓨터공학부 교수(kjhan@db.konkuk.ac.kr)

moving objects increases, and the number of time division increases.

주요어 : 시공간데이타베이스, 이동 객체, 이동 패턴

Keyword : Spatio-Temporal Database, Moving Object, Moving Pattern

1. 서 론

최근 들어 이동 객체의 이력 데이타에서 이동 객체의 이동 패턴, 즉 연속되는 시간 영역에서 반복적으로 발생되는 공간 이동 경로와 같은 다양한 지식을 추출하여 활용하는 응용 서비스의 활용성이 점점 증대되고 있다. 특히, 이러한 응용 서비스에서는 이동 객체의 이동 패턴을 이용하여 고객의 이동 특성에 따라 개인화(personalization) 함으로써 알맞은 컨텐츠나 서비스 제공을 가능하게 할 수 있으며[1], 이동 패턴을 이용하여 이동 객체의 미래 위치를 예측함으로써 미래의 교통 변화 상황을 고려한 교통 통제 및 이동 경로 안내 서비스 등을 제공할 수 있다.

시공간 데이타 마이닝(spatio-temporal data mining)은 이미 수집된 방대한 양의 시공간 데이타 집합, 즉 이력 데이타로부터 이동 객체의 이동 패턴 정보 등을 빠르고 정확하게 추출하는 연구 분야이다. 그러나 현재 시공간 데이타 마이닝에서 대부분의 이동 객체의 이동 패턴 추출에 관한 연구는 시간을 고려하지 않은 공간상의 이동 패턴을 추출하는데 그치고 있다. 예를 들어, 16시에서 18시 사이 이동 객체의 이동 패턴을 추출한다고 가정한다면, "자양동→천호동→잠실동"과 같은 공간 이동 패턴을 추출하는 것이다. 따라서 16:00(자양동)→16:30(천호동)→17:00(천호동)→17:30(잠실동)→18:00(잠실동)과 같이 시간 정보를 갖는 공간, 즉 위치가 이동하는 이동 패턴을 추출하는 것에 대한 연구가 필요하다.

최근 들어 일반적인 패턴 탐사 방법을 확장한 이동 패턴 추출 방법들이 연구되고 있다. STPMine1, STPMine2 알고리즘은 주기적으로 빈발하게 발생되는 이동 패턴과 부분 이동 패턴 추출을 효율적으로 수행하기 위한 이동 패턴 추출 방법이다[2,3]. 그러나 이동 패턴으로서 의미가 있는 최소 발생빈도인 최소지지도가 낮은 경우에 많은 수의 후보 이동 패턴이 생성되고 이로 인하여 수행 시간과 소요 메모리가 급격히 증가하게 되는 단점이 있다.

따라서 본 논문에서는 대용량의 시공간 데이타 집합으로부터 이동 객체의 이동 패턴을 효율적으로 추출하기 위한 STMPE 알고리즘을 제안한다. STMPE 알고리즘은 이동 객체의 이력 정보를 분석하여 시공간 정보를 나타내는 일반화된 이동 패턴을 추출하기 위한 알고리즘이다. STMPE 알고리즘은 시공간 데이타 일반화 기능, 단기 이동 패턴 추출 기능, 이동 패턴 트리 생성 기능 등을 지원한다.

본 논문에서 제안한 STMPE 알고리즘은 시공간 데이타를 일반화시킴으로서 대용량 시공간 데이타 저장으로 인한 메모리 사용량을 최소화할 수 있으며, 데이타베이스 스캔시 단기 이동 패턴을 추출하여 이용함으로서 데이타베이스 스캔 횟수를 최소화하여 수행 시간을 단축할 수 있다. 또한 STMPE 알고리즘은 수행 시간이 짧고 소요 메모리의 양이 적기 때문에 최소지지도가 낮은 시공간 이동 패턴을 추출하는 경우에도 효율적으로 적용이 가능하다. 그래서 대용량의 시공간 데이타에서 최소지지도가 높거나 낮은 모든 경우에 대하여 수행 시간 및 메모리 공간의 비용이 비슷하게 소요되므로 효율적인 시공간 이동 패턴의 추출이 가능하다.

본 논문의 구성은 다음과 같다. 2장에서는 관련 연구를 설명하고, 3장에서는 본 논문에서 제시한 STMPE 알고리즘에 대하여 설명한다. 4장에서는 기존의 이동 패턴 추출 알고리즘과 STMPE 알고리즘의 성능평가에 대하여 기술한다. 그리고 마지막으로 제 5 장에서는 결론에 대해 언급한다.

2. 관련 연구

이동 패턴 추출 알고리즘은 공간 이동 패턴 추출과 시공간 이동 패턴 추출로 나눌 수 있다. 공간 이동 패턴 추출 알고리즘은 시간의 흐름에 따라 공간을 축으로 하여 이동 객체의 이동 패턴을 추출하는 방법이다. SPADE 알고리즘은 간단한 조인과 효율적인 격자 검색 기술을 사용하여 빈발 시퀀스를 찾기 위한 알고리즘이다[4]. SPADE 알고리즘은 이동 패턴 추출에 적용할 수는 있으나 이동 객체의 이동 패턴 보다는 자연 현상에 대한 패턴을 분석하는 알고리즘으로 적합하기 때문에 시공간 이동 패턴 추출에는 효율적이지 못하다.

시공간 이동 패턴 추출 알고리즘은 이동 객체의 이력 데이타를 시간과 공간의 공동 축으로 하여 위치가 변화함에 따라 발생되는 이동 패턴을 추출하는 방법이다. STPMine1 알고리즘[3]은 시공간 이동 객체들의 시간에 따른 이동 패턴을 추출하는 알고리즘이다. STPMine1 알고리즘은 이동 객체의 이력 데이타를 각 시간 영역 단위 별로 사용자가 미리 정의한 공간 영역에 대하여 1차 빈발 패턴들을 발견하고 그 결과를 입력으로 사용한다. 또한, STPMine1 알고리즘은 연관 규칙 AprioriTID 알고리즘[5]을 변형하여 빈발 이동 패턴을 추출할 수 있다. 이 기법은 AprioriTID 알고리즘이 패턴의 길이만큼 데이타베이스를 반복 스캔하는 문제점 때문에 이동 패턴 추출 알고리즘에서 시간 영역의 수(즉, 이동 패턴의 길이)가 증가할수록 수행시간과 소요 메모리의 양이 급격하게 증가하는 단점이 있다.

STPMine2 알고리즘[3]은 시간과 공간을 축으로 하는 이동 객체의 이동 패턴을 추출하는 알고리즘이며 STPMine1 알고리즘과는 달리 데이타베이스 읽는 횟수를 2회로 줄여서 이동 패턴을 구할 수 있다. 이것은 이동 패턴을 추출하는 데이타 마이닝 수행시간을 줄일 수 있으며, 최소지지도가 높은 이동 패턴 추출과 숨겨져 있는 부분 패턴을 추출하는데 이용될 수 있다. 그러나, STPMine2 알고리즘은 데이타를 읽는 횟수는 줄였지만 1차 빈발 영역으로 작성한 Super 패턴과 Shadow 패턴이 최소지지도가 낮을수록, 이동 객체의 수가 증가 할수록, 시간 영역의 수가 늘어날수록 필요한 메모리의 양이 기하급수적으로 증가하고, 후보 패턴 수가 기하급수적으로 많아져서 비교 연산의 횟수가 증가하기 때문에 시스템에 부하가 많이 걸리는 단점이 있다.

이러한 단점들을 극복하기 위해 이동 패턴 추출시의 수행시간 및 소요 메모리의 양을 최소화할 수 있는 이동 패턴 추출 알고리즘이 필요하다. 그리고 데이타베이스 스캔 횟수와 이동 패턴 생성 및 검증 시간을 최소화하는 방법과 분석에 필요한 이력 데이타나 후보 이동 패턴을 저장할 때 소요되는 메모리의 양을 최소화하는 방법이 필요하다. 또한, 최소지지도의 높고 낮음, 데이타의 많고 적음에 관계없이 일정한 수행성능을 유지하는 이동 패턴 추출 방법이 필요하다.

본 논문에서는 이를 위하여 대용량 시공간 데이타의 저장을 위한 메모리 사용량을 최소화하고, 데이타베이스 스캔 횟수를 최소화하여 수행시간을 단축하는 새로운 STMPE 알고리즘을 제시하였다.

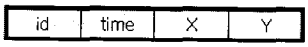
3. STMPE 알고리즘

본 장에서는 STMPE 알고리즘에서 사용되는 자료구조와 STMPE 알고리즘의 구성에 대해 설명한다.

3.1 STMPE 자료구조

3.1.1 STD(Spatio-Temporal Data)

STD는 시공간 데이타베이스에서 추출한 이동 객체의 이력 데이타를 저장하는 자료 구조이다. 그림 1은 STD의 자료 구조이다.

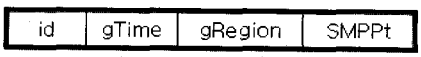


<그림 1> STD 자료 구조

STD는 이동 객체를 식별하기 위한 id, 위치 데이타를 획득한 시간 정보를 저장하기 위한 time, 이동 객체의 위치 데이타를 표현하는 점의 좌표를 저장하기 위한 X, Y로 구성된다.

3.1.2 GSTD(Generalized Spatio-Tem-poral Data)

GSTD는 STD에 저장된 이동 객체의 이력 데이타를 일반화한 값을 저장하기 위한 자료 구조이다. 그림 2는 GSTD의 자료 구조를 보여준다..



<그림 2> GSTD 자료 구조

GSTD는 이동 객체를 식별하기 위한 id, 이동 객체의 시간을 일반화한 값을 저장하기 위한 gTime, 이동 객체의 위치 데이타를 일반화한 값을 저장하기 위한 gRegion, 해당 id의 이전 단기 이동 패턴 정보를 저장하고 있는 SMP의 주소를 저장하기 위한 SMPPt로 구성된다. SMPPt는 NULL 값을 갖거나 해당 id가 저장된 SMP의 주소를 저장한다. 이동 패턴을 추출할 때 SMPPt를 통해 해당 id가 속한 이전 단기 이동 패턴을 저장하고 있는 SMP를 직접 접근함으로써 이동 객체의 단기 이동 패턴 검색의 효율성이 증가된다.

3.1.3 SMP(Short Moving Pattern)

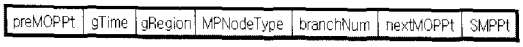
SMP는 단기 이동 패턴의 정보를 저장하기 위한 자료 구조이다. 그림 3은 SMP의 자료 구조이다.

<그림 3> SMP 자료 구조

SMP는 단기 이동 패턴을 구성하는 이전 시간 영역의 일반화된 공간 영역을 저장하기 위한 preRegion, 현재 시간 영역의 일반화된 공간 영역을 저장하기 위한 curRegion, 현재 SMP에 등록된 이동 객체들의 수를 저장하기 위한 FC, 이전 시간 영역에 등록되었던 이동 객체의 id 리스트를 저장하기 위한 oldIDList, 현재 시간 영역에 등록된 이동 객체의 id 리스트를 저장하기 위한 newIDList, 현재 단기 이동 패턴을 저장 중인 SMP들을 연결 리스트 구조로 구성하기 위해서 현재의 SMP를 기준으로 각각 이전 SMP와 다음 SMP를 연결하기 위해 SMP 주소를 저장하기 위한 prevSMPPt와 nextSMPPt, SMP와 연관된, 즉 SMP를 이동 패턴의 요소로 가지는 MOP 객체의 주소를 저장하기 위한 mopPt, 현재의 SMP에 등록된 id들의 GSTD 주소를 저장하기 위한 gLink로 구성된다.

3.1.4 MOP(Moving Pattern)

MOP는 장기 이동 패턴을 저장하기 위한 최소 단위의 자료 구조이다. 그림 4는 MOP의 자료 구조를 보여준다.



<그림 4> MOP 자료 구조

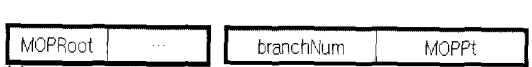
MOP는 시간을 일반화한 값을 저장하기 위한 gTime, 일반화된 공간 영역 값을 저장하기 위한 gRegion, MOP의 연결 타입에 대한 값을 저장하기 위한 nextNodeType, MOP의 자식 MOP의 수를 저장하기 위한 branchNum, MOP와 연결된 이전 MOP에 대한 주소를 저장하기 위한 preMOPPt, 연결된 다음 MOP 주소를 저장하기 위한 nextMOPPt, MOP에 연결되는 SMPPt로 구성된다. nextNodeType은 MOP의 연결 상태를 분류하며 표 1과 같은 값을 가진다.

<표 1> MOP 분류를 위한 nextNodeType

nextNodeType	nextMOPPt 값	SMPPt 값	상 태
0	다음 MOP 주소	null	중간노드
1	null	다음 SMP 주소	진행상태
2	null	마지막 SMP 주소	완료상태

3.1.5 PatternTree

패턴 트리는 각 시간에 따른 장기 이동 패턴들을 갖는 트리를 말한다. PatternTree는 최대로 시간 영역의 수만큼의 MOPRoot를 갖는다. 그림 5는 PatternTree와 MOPRoot의 자료 구조를 보여준다.



<그림 5> 패턴 트리와 MOPRoot의 자료 구조

MOPRoot는 자식 MOP의 수를 저장하기 위한 branchNum과 자식 MOP의 주소를 갖는 rootMOPPt로 구성된다. MOPRoot는 임의의 시간 영역에서 시작하는 이동 패턴의 시작점 역할을 한다. 또한 MOPRoot는 각 시간 영역에서 새롭게 시작하는 이동 패턴의 시작점 역할을 하는 MOP가 여러 개 나타날 수 있기 때문에 배열 형태로 여러 개의 MOP 객체 주소를 가질 수 있게 되어있다.

PatternTree는 (최대 시간 영역의 수 - 1) 만큼의 MOPRoot의 주소를 mopRootPt에 저장한다. PatternTree는 각 시간 영역에서 새롭게 시작하는 이동 패턴의 정보를 가지고 있는 MOPRoot 객체를 가지므로 전체 시간 영역에 해당하는 MOPRoot로 전체 이동 패턴 트리를 구성한다.

3.2 STMPE 알고리즘의 구성

본 장에서는 STMPE 전체 알고리즘과 알고리즘을 구성하고 있는 시공간 데이타 일반화, 단기 이동 패턴 추출, 그리고 이동 패턴 트리 생성 등의 알고리즘을 설명한다.

전체 STMPE 알고리즘은 그림 6과 같다. STMPE 알고리즘의 입력은 이동 객체 이력 데이타 std, 이동 객체의 수 IDLength, 시간 분할 횟수 timeLength 등이 있으며, 출력으로는 시공간 이동 패턴을 가지고 있는 PatternTree pt를 반환한다. STMPE 알고리즘은 입력된 이동 객체의 이력 데이타를 STMPE 알고리즘에서 사용할 수 있는 std에 저장하기 위한 GetData() 함수, std에 저장되어 있는 시공간 데이타를 일반화된 시공간 데이타로 변환하여 gstd에 저장하기 위한 GeneralSTD() 함수, 단기 이동 패턴을 생성하고 지지도를 계산하는 InitSMP() 함수와 AddSMP() 함수, 각 시간 영역에서 단기 이동 패턴이 최소지지도를 만족하는지를 비교하여 이동 패턴 트리에 등록하는 Make

MOP() 함수 등으로 구성된다.

```
STMPE()
Input : the number of moving object IDLength, the
        number of time region timeLength
Output : moving pattern tree pt
std := null; // std is spatio-temporal data
gstd := null; // gstd is generalized spatio-temporal data
smpHd := null; // smpHd is head of short moving
          pattern list
pt := null; // pt is moving pattern tree
for t:=1 to timeLength do
  std := GetData(std, t, IDLength);
  gstd := GeneralSTD(std, gstd);
  if t = 1 then
     smpHd := InitSMP(&gstd, smpHd);
  else
     smpHd := AddSMP(&gstd, smpHd);
  end if
  pt := MakeMOP(&smpHd, &gstd, pt);
end for
return pt;
```

<그림 6> 전체 STMPE 알고리즘

3.2.1 시공간 데이타 일반화

시공간 데이타 일반화는 이동 객체의 위치 및 시간 데이타를 사용자의 정의에 따라 자동 생성되는 일반화 값으로 변환한다. 이로 인하여 데이타를 표현하는데 필요한 공간의 크기를 줄일 수 있으며, 시공간의 특성을 이용한 부가가치가 높은 새로운 정보, 즉 이동 패턴을 추출할 수 있다. STMPE 알고리즘에서는 대용량의 시공간 데이타를 일반화하여 전체 데이타의 크기를 줄일 수 있다. 시공간 데이타를 일반화하는 공식은 다음 그림 7과 같다.

$$id_n\text{의 시간 일반화 값} = \left(\frac{(\text{시간} - \text{가장 최초시간})}{\left(\frac{(\text{가장 마지막시간} - \text{가장 최초시간})}{\text{시간분할횟수}-1}\right)}\right)+1$$

(a) 시간 일반화 공식

$$id_n\text{의 X 좌표 일반화 값} = \left(\frac{X\text{좌표값}}{\left(\frac{(\text{영역의 최대}X\text{값} - \text{영역의 최소}X\text{값})}{\text{영역분할값}}\right)}\right)+1$$

$$id_n\text{의 Y 좌표 일반화 값} = \left(\frac{Y\text{좌표값}}{\left(\frac{(\text{영역의 최대}Y\text{값} - \text{영역의 최소}Y\text{값})}{\text{영역분할값}}\right)}\right)+1$$

(b) 공간 일반화 공식

<그림 7> 시공간 일반화 공식

시공간 데이타 일반화 알고리즘은 그림 8과 같다. 알고리즘의 입력 부분은 std, gstd 등이 있으며, 결과로 gstd를 반환한다. 또한 이동 객체의 데이타 일반화를 위해서 시간의 일반화를 위한 GeneralTime() 함수, 공간의 일반화를 위한 GeneralRegion() 함수, 일반화된 시공간 데이타를 gstd에 저장하기 위한 SETGSTD() 함수가 있다.

```
GeneralSTD()
Input : spatio-temporal data std, generalized
        spatio-temporal data gstd, time region t
Output : generalized spatio-temporal data gstd
for t:=1 to idLength do
  tstd := std[t];
  gTime := GeneralTime(tstd.time);
  gRegion := GeneralRegion(tstd.X, tstd.Y);
  SetGSTD(gstd, tstd.id, gTime, gRegion);
end for
return gstd;
```

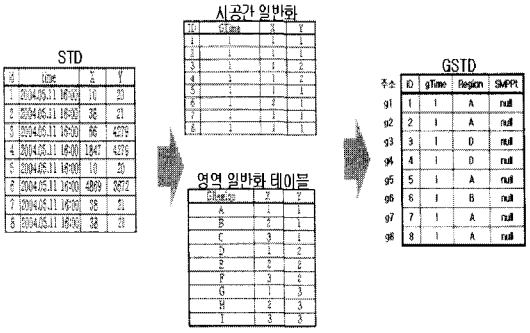
<그림 8> 시공간 일반화 알고리즘

그림 9는 STMPE 알고리즘에서 일반화하는 과정으로 그림 3의 공식을 이용하여 특정 시간 t1에 해당하는 STD를 GSTD로 변환하는 예를 보여 주고 있다.

3.2.2 단기 이동 패턴 추출

단기 이동 패턴 추출은 일반화된 시간 영역에서 연속적인 두 개의 시간 영역에 대한 이동 패턴, 즉 단기 이동 패턴을 추출한다. 추출된 단기 이동 패턴은 이동 패턴 트리를

구성하는 최소 이동 패턴이다. 또한 각 시간 영역 별로 단기 이동 패턴 추출 연산을 수행하기 때문에 대용량의 시공간 데이타베이스를 시간 영역 별로 나누어 입력할 수 있다. 단기 이동 패턴 추출 알고리즘은 이동 객체의 이력 데이타를 대상으로 단기 이동 패턴을 추출하기 위해 처음 시작하는 시간 영역에서 실행되는 InitSMP 알고리즘과 나머지의 시간 영역에서 실행되는 AddSMP 알고리즘이 있다.



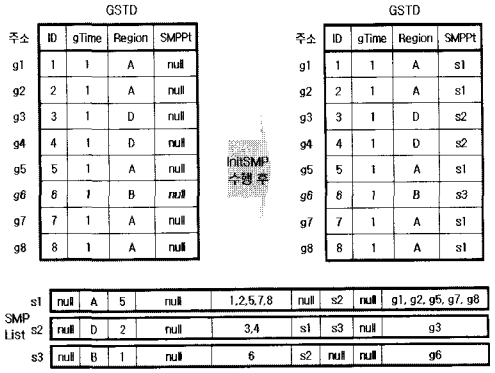
<그림 9> STD에서 GSTD로 변환 예

InitSMP 알고리즘은 그림 10과 같다. 알고리즘의 입력 부분은 gstd, 단기 이동 패턴 리스트의 헤드 정보를 가지고 있는 smpHd 등이 있다. 그리고 출력 부분은 생성된 단기 이동 패턴 리스트의 헤드 정보를 가지고 있는 smpHd를 반환한다. 또한 단기 이동 패턴이 존재하지 않을 때, 즉 단기 이동 패턴 리스트의 헤드 값이 NULL일 때 단기 이동 패턴을 생성하는 CreateSMP() 함수, 생성된 단기 이동 패턴을 단기 이동 패턴 리스트에 추가하는 AddLink() 함수, 생성된 단기 이동 패턴 리스트 중에서 이동 객체의 일반화된 공간 영역과 동일한 공간 영역을 갖는 단기 이동 패턴을 찾는 FindSMP() 함수, 찾은 단기 이동 패턴에 이동 객체의 정보를 등록하는 AddEntry() 함수가 있다.

```
InitSMP()
Input : generalized spatio-temporal data gstd, head of
        short moving pattern list smpHd
Output : head of short moving pattern list smpHd
tsmp := CreateSMP(gstd[1]);
smpHd := AddLink(smpHd, tsmp);
AddEntry(tsmp, gstd[1]);
for i:=2 to IdLength do
  tgstd := gstd[i];
  tsmp := FindSMP(tgstd->gRegion);
  if tsmp = NULL then
     tsmp := CreateSMP(tgstd);
     smpHd := AddLink(smpHd, tsmp);
  end if
  AddEntry(tsmp, tgstd);
end for
return smpHd;
```

<그림 10> 단기 이동 패턴 추출 알고리즘 InitSMP

그림 11은 InitSMP 단기 이동 패턴 추출을 수행함에 따른 결과를 예제로 보여준다. 입력된 8개의 이동 객체에 대하여 gstd의 SMPPt가 NULL에서 InitSMP 단기 이동 패턴 추출 과정을 수행한 후 해당 SMP 주소를 갖는 결과를 보여준다. 그리고 InitSMP 단기 이동 패턴 추출 과정에서 생성된 단기 이동 패턴 SMP 리스트를 보여 주고 있다.



<그림 11> InitSMP 단기 이동 패턴 추출 예제

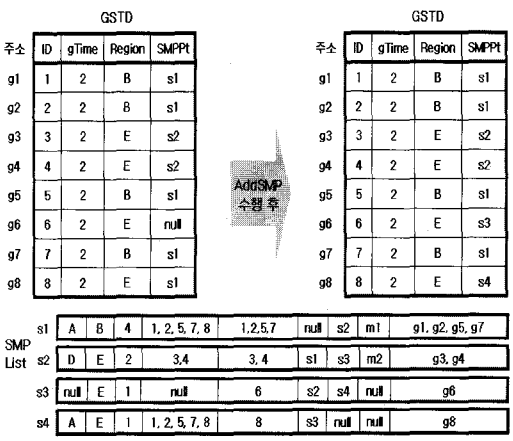
AddSMP 알고리즘은 그림 12와 같다. 알고리즘의 입력 부분은 이동 객체의 이력 데이타를 일반화한 정보를 가지는 gstd, 단기 이동 패턴 리스트의 헤드 정보를 가지고 있는 smpHd가 있다. 그리고 출력 부분은 생성된 단기 이동 패턴 리스트의 헤드 정보를 가지고 있는 smpHd를 반환한다.

또한 생성된 단기 이동 패턴 리스트 중에서 이동 객체의 공간 영역과 동일한 값을 갖는 단기 이동 패턴을 검색하는 FindSMP() 함수, 해당되는 공간 영역을 갖는 단기 이동 패턴이 존재하지 않을 때, 또는 단기 이동 패턴 리스트의 헤드 값이 NULL일 때 단기 이동 패턴을 생성하는 CreateSMP() 함수, 생성된 단기 이동 패턴을 단기 이동 패턴 리스트에 추가하는 AddLink() 함수, 검색된 단기 이동 패턴에 이동 객체의 정보를 등록하는 AddEntry() 함수, 이전 시간 영역에서 공간 영역의 값이 같고 현재 시간 영역에서 공간 영역의 값이 동일한 단기 이동 패턴을 검색하는 FindLinkSMP() 함수가 있다.

```
AddSMP()
Input  : generalized spatio-temporal data gstd,
         head of short moving pattern smpHd
Output : head of short moving pattern smpHd
for t :=1 in idLength do
  tgstd := gstd[t];
  if tgstd.SMPPt = NULL then
     tsmp := FindSMP(tgstd->gRegion);
     if tsmp = NULL then
        tsmp := CreateSMP(tgstd);
        smpHd := AddLink(smpHd, tsmp);
     end if
     AddEntry(tsmp, tgstd);
  else
     tsmp := tgstd.SMPPt;
     if tsmp.curRegion != tgstd.gRegion then
        tsmp := FindLinkSMP(tgstd->gRegion);
        if tsmp = NULL then
              tsmp := CreateSMP(tgstd);
              smpHd := AddLink(smpHd, tsmp);
        end if
     end if
     AddEntry(tsmp, tgstd);
  end if
end for
return smpHd;
```

<그림 12> 단기 이동 패턴 추출 알고리즘 AddSMP

그림 13을 보면 그림 11과 달리 입력된 gstd.SMPPt의 값 모두가 null만 있는 것은 아니다. 현재의 이동 객체가 이전 시간 영역에 최소지지도를 만족하는 단기 이동 패턴 SMP에 속했다면 해당 SMP의 주소를 갖게 되며, 그렇지 않고 해당 단기 이동 패턴이 최소지지도를 만족하지 못했다면 이동 객체 g6과 같이 SMPPt의 값이 null로 초기화 되어 있는 것을 보여 주고 있다.



<그림 13> AddSMP 단기 이동 패턴 추출 예제

3.2.3 이동 패턴 트리 생성

이동 패턴 트리 생성 알고리즘은 단기 이동 패턴 추출 알고리즘에서 생성된 단기 이동 패턴 리스트들을 사용자가 정의한 최소지지도를 만족하는지를 비교하고, 만족하는 단기 이동 패턴을 MOP로 생성하여 이동 패턴 트리에 등록하는 알고리즘이다. 이렇게 구축된 이동 패턴 트리는 시공간 이동 패턴이 필요한 실세계의 다양한 응용 서비스에서 활용될 수 있다.

이동 패턴 트리 생성 알고리즘은 그림 14과 같다. 이 알고리즘의 입력 부분은 이동 패턴 정보를 가지고 있는 pt, 단기 이동 패턴 리스트의 헤드 정보를 가지고 있는 smpHd,

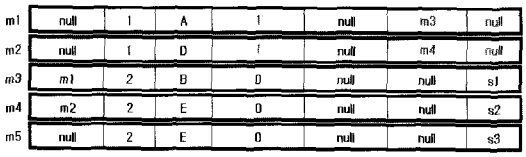
일반화된 시공간 데이타 gstd가 있다.

```
MakeMOP()
Input : head of short moving pattern smpHd, generalized
        spatio-temporal data gstd, moving pattern tree pt
Output : moving pattern tree pt
tsmp := smpHd;
while(tsmp != null) do
  if tsmp.FC ≥ supportMin then
    pt := MakeNewMOP(pt, tsmp);
    MoveSMP(tsmp);
  end if
  tsmp := tsmp.nextSMPPt;
end while
AggregateSMP(smpHd);
tsmp := smpHd;
while(tsmp != null) do
  if tsmp.FC ≥ supportMin then
    pt := MakeNewMOP(pt, tsmp);
    MoveSMP(tsmp);
  else
    DeleteSMP(tsmp);
  end if
  tsmp := tsmp.nextSMPPt;
end while
return pt;
```

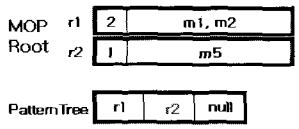
<그림 14> 이동 패턴 트리 생성 알고리즘

그리고 출력 부분은 이동 패턴 정보를 가지고 있는 pt를 반환한다. 이 알고리즘은 최소지지도를 만족하는 단기 이동 패턴을 대상으로 MOP를 생성하고 생성된 MOP를 이동 패턴 트리에 등록하는 MakeNewMOP() 함수, MOP를 생성한 단기 이동 패턴을 다음 시간 영역에서 사용할 수 있게 초기화하는 MoveSMP() 함수, 최소지지도를 만족하지 못하는 단기 이동 패턴들을 대상으로 공통된 공간 영역을 기준으로 통합하는 AggregateSMP() 함수, 최소지지도를 만족하지 못하는 단기 이동 패턴을 삭제하는 DeleteSMP() 함수를 사용한다.

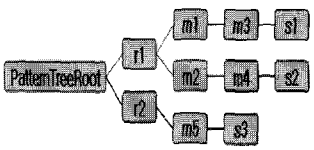
그림 15는 생성된 MOP 리스트 (a), MOPRoot 및 PatternTree 자료 구조 구성 (b), 그리고 생성된 이동 패턴 트리를 트리 형태로 표현한 (c)를 보여 주고 있다. 최소지지도를 만족하는 단기 이동 패턴을 그림 15(a)와 같이 시간 영역 2에서 새롭게 시작하는 이동 패턴 MOP m5를 생성하게 되면, 그림 15(b)와 같이 MOPRoot r2에 m5를 등록하게 되며, 이것은 새롭게 시간 영역 2에서 이동 패턴이 시작되었기 때문에 PatternTree에 r2를 등록하게 된다. 이러한 과정을 거치면서 이동 패턴 트리가 구성되며 이것을 트리 구조로 알기 쉽게 표현한 것이 그림 15(c)이다.



(a) 생성된 MOP 리스트



(b) MOPRoot 및 PatternTree 자료 구조 구성



(c) Pattern Tree 생성

<그림 15> 생성된 MOP 리스트와 MOPRoot 및 패턴 트리 구성 예제

4. 성능 평가

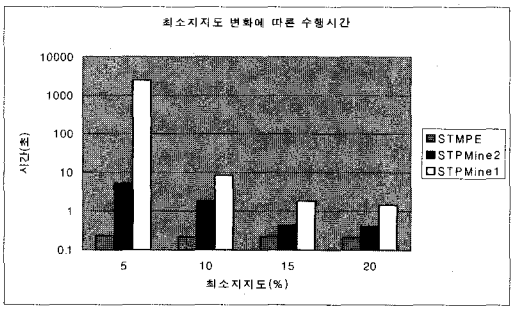
STMPE 알고리즘의 이동 패턴 추출 성능 평가에서 사용된 이동 객체의 이력 데이타는 네트워크 기반 위치 데이타 생성기를 이용하여 생성하였다. 네트워크 기반 위치 데이타 생성기에서 이동 객체의 다양한 이력 데이타를 생성하기 위해서 위치 데이타의

생성 조건으로 이동 객체의 이동 속도, 이동 객체의 수, 시간 분할 횟수 등을 다양하게 입력하였다. STMPE 알고리즘의 이동 패턴 추출 성능 평가를 위해서 생성한 이력 데이타의 생성 조건은 다음과 같다. 이동 객체의 이동 속도는 빠름, 중간, 느림으로 변화시켜 생성하였다. 그리고, 이동 객체의 수는 200개, 400개, 600개, 800개, 1000개의 경우를 생성하였고, 이동 객체에 대한 위치의 보고 횟수는 2, 4, 6, 8, 10회의 경우로 각각 생성하였다.

본 성능 평가에서는 기존의 이동 패턴 추출 알고리즘(즉, STPMine1과 STPMine2)과 STMPE 알고리즘의 수행시간과 소요 메모리의 양에 대한 비교 평가를 하기 위하여 이동 객체의 수, 시간 분할 횟수, 최소지지도 등의 세 가지 변수 중 두 가지 변수를 고정하고 나머지 한 가지 변수의 값을 변화시키면서 실험한 결과를 비교하였다.

4.1 수행시간 비교 평가

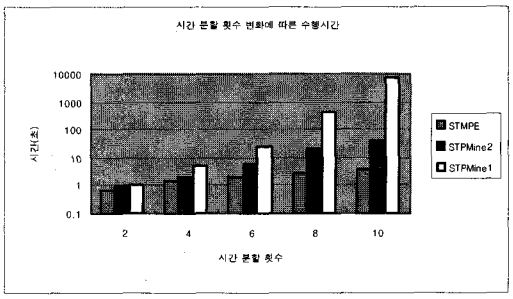
그림 16은 STMPE 알고리즘과 기존의 알고리즘과의 최소지지도 변화에 따른 수행시간을 비교한 그래프이며, 이동 객체의 수 200개, 시간 분할 횟수는 4로 고정하고 최소지지도를 5%, 10%, 15%, 20%로 변화시킬 경우 시공간 이동 패턴 추출에 걸리는 수행 시간을 보여준다.



<그림 16> 최소지지도 변화에 따른 수행시간 비교

그림 16에서 보는 바와 같이 STMPE 알고리즘은 최소지지도 변화와는 무관하게 일정한 수행 시간을 보이고 있으나 기존의 알고리즘은 최소지지도가 낮아짐에 따른 후보 패턴의 수가 급격하게 증가하게 되고, 이에 따라 비교 연산의 횟수가 늘어나기 때문에 수행 시간이 급격히 증가함을 알 수 있다. STMPE 알고리즘이 최소지지도의 변화에 영향을 받지 않은 이유는 각 시간 영역 별로 유효한 시공간 이동 패턴만을 남기고 나머지는 삭제가 되기 때문에 후보 시공간 이동 패턴의 수가 상대적으로 훨씬 적게 유지되기 때문이다.

그림 17은 STMPE 알고리즘과 기존의 알고리즘과의 시간 분할 횟수의 변화에 따른 수행 시간을 비교한 그래프이며, 이동 객체의 수 200개, 최소지지도는 20%로 고정하고 시간 분할 횟수를 2회, 4회, 6회, 8회, 10회로 변화시킬 경우 시공간 이동 패턴 추출에 걸리는 수행 시간을 보여준다.

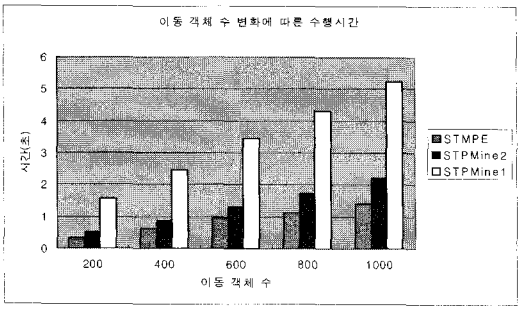


<그림 17> 시간 분할 횟수 변화에 따른 수행시간 비교

이러한 성능 비교는 시간 분할 횟수가 길어짐에 따라, 즉 시간 영역의 수가 증가함에 따라 수행 시간에 미치는 영향이 얼마나 되는지를 살펴보기 위한 실험이다. 기본적으로 시간 영역이 길어지게 되면 동일한 이동 객체의 수에 대한 시공간 이동 패턴 추출이므로 데이타의 양이 계속 증가하게 된다. 즉, 데이타 I/O 시간이 계속해서 늘어나는 것이

다. 그림 17에서 보는 바와 같이 STP Mine1, STPMine2와 STMPE 알고리즘 모두가 시간 분할 횟수가 증가함에 따라 수행시간이 늘어나고 있는 모습을 보이고 있다. 그 이유는 기본으로 I/O 시간의 증가가 있고, 또한 세 알고리즘 모두가 시간 정보를 갖는 부분 시공간 이동 패턴을 추출하기 때문에 후보 패턴의 수가 기하급수적으로 증가하기 때문이다. 그러나 STMPE 알고리즘은 STPMine1, STPMine2 알고리즘 보다 상대적으로 적은 증가율을 보이고 있다. 즉, STMPE 알고리즘은 데이타 I/O가 한 번 수행되기 때문에 더 나은 성능을 보이고 있다.

그림 18은 STMPE 알고리즘과 기존의 알고리즘과의 이동 객체 수의 변화에 따른 수행 시간을 비교한 그래프이며, 최소지지도 20%, 시간 분할 횟수는 4회로 고정하고 이동 객체의 수를 200개, 400개, 600개, 800개, 1000개로 변화시킬 경우 시공간 이동 패턴 추출에 걸리는 수행 시간을 보여준다.

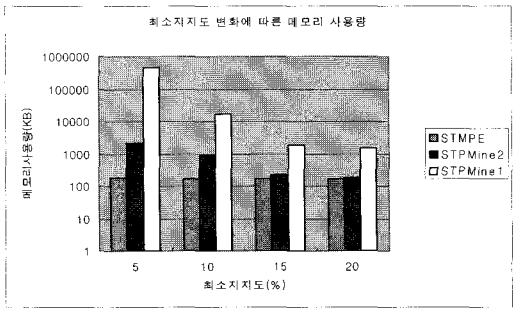


<그림 18> 이동 객체 수 변화에 따른 수행시간 비교

이동 객체의 수가 증가함에 따라 수행 시간에 영향을 미치는 부분은 두 가지가 있다. 첫째는 입력되는 이력 데이타가 커짐에 따른 I/O 시간의 증가이고, 둘째는 후보 시공간 이동 패턴의 수가 많아짐에 따른 비교 연산 횟수의 증가이다. 이 두 가지의 영향으로 전체적인 수행 시간이 증가하는 결과를 얻게 된다. 그림 18에서 보는 바와 같이 이동 객체 수의 증가에 따라 모든 알고리즘의 수행 시간이 증가하고 있다. 그러나 그림 18의 이동 객체 수의 증가에 따른 수행 시간 증가보다 그림 17의 시간 분할 횟수의 증가에 따른 수행 시간 증가가 상대적으로 훨씬 높게 나타난다. 그 이유는 3×3의 그리드 형태 분할을 기준으로 한다면, 시간 분할 횟수를 T라고 했을 때 후보 시공간 이동 패턴의 수가 $(3\times3)^T$개의 형태로 기하급수적으로 증가하기 때문에 시간 분할 횟수의 증가에 따른 비용의 증가가 상대적으로 심하게 나타나게 된다.

4.2 소요 메모리의 양 평가

그림 19는 STMPE 알고리즘과 기존의 알고리즘과의 최소지지도 변화에 따른 소요 메모리의 양을 비교한 그래프이며, 이동 객체 수 200개, 시간 분할 횟수는 4회로 고정하고 최소지지도를 5%, 10%, 15%, 20%로 변화시킬 경우 시공간 이동 패턴 추출에 필요한 소요 메모리의 양을 보여준다.

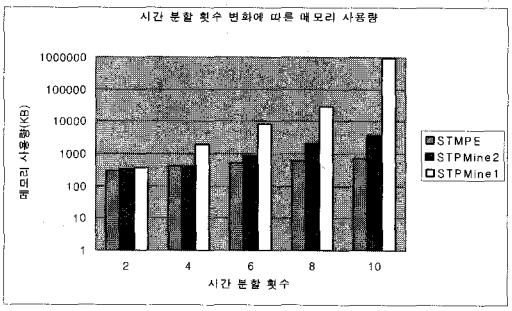


<그림 19> 최소지지도 변화에 따른 메모리의 사용량 비교

이 실험을 통해 STMPE 알고리즘, STP Mine1 알고리즘, STPMine2 알고리즘에서 최소지지도의 변화에 따라서 발견되는 후보 시공간 이동 패턴 수의 변화가 마이닝 과정에서 전체 소요 메모리 양에 미치는 영향이

어느 정도인지를 판단할 수 있다. 그림 19에서 보는 바와 같이 STMPE 알고리즘이 상대적으로 훨씬 소요 메모리의 양이 적다는 것을 볼 수 있다. STPMine1 알고리즘의 경우에는 최소지지도가 5%인 경우 메모리의 사용량이 시스템이 메모리 임계치에 도달하는 결과를 보여주고 있다. 메모리의 양이 많이 필요하게 되면 상대적으로 메모리의 스와핑에 의한 I/O 시간의 증가로 수행 시간이 많이 걸리게 된다. 본 논문에서 제안한 STMPE 알고리즘은 최소지지도의 높고 낮음에 따른 소요 메모리의 양의 변화가 거의 없게 나타나고 있다. 일반적으로 최소지지도가 낮아짐에 따라 수행 시간이나 메모리의 양에 대한 비용이 증가하는 것에 비해 STMPE 알고리즘은 최소지지도 변화에 따른 비용의 변화가 거의 없기 때문에 최소지지도가 낮은 시공간 이동 패턴 추출에도 강한 장점을 가지게 된다.

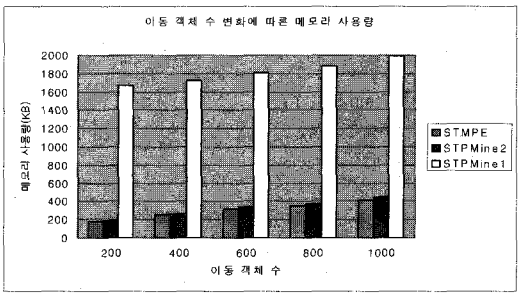
그림 20은 STMPE 알고리즘과 기존의 알고리즘과의 시간 분할 횟수 변화에 따른 소요 메모리의 양을 비교한 그래프이며, 이동 객체 수 200개, 최소지지도는 20%로 고정하고 시간 분할 횟수를 2회, 4회, 6회, 8회, 10회로 변화시킬 경우 시공간 이동 패턴 추출에 필요한 소요 메모리의 양을 보여준다.



<그림 20> 시간 분할 횟수 변화에 따른 메모리의 사용량 비교

그림 20에서 보는 바와 같이 STMPE 알고리즘은 시간 분할 횟수가 증가함에 따라 메모리의 양이 점진적으로 약간씩 증가하고 있는 모습을 보이고 있다. 이것은 STMPE 알고리즘이 시간 분할 횟수가 많은, 즉 추출하는 최대 시공간 이동 패턴의 길이가 길어진다고 해도 안정적으로 수행될 수 있는 것을 나타낸다. 이에 비해 다른 알고리즘은 메모리의 양이 기하급수적으로 증가하고 있다.

그림 21은 STMPE 알고리즘과 기존의 알고리즘과의 이동 객체 수 변화에 따른 소요 메모리의 양을 비교한 그래프이며, 시간 분할 횟수 4회, 최소지지도는 20%로 고정하고 이동 객체 수를 200개, 400개, 600개, 800개, 1000개로 변화시킬 경우 시공간 이동 패턴 추출에 필요한 소요 메모리의 양을 보여준다.



<그림 21> 이동 객체 수 변화에 따른 메모리의 사용량 비교

그림 21에서 보는 바와 같이 STMPE 알고리즘이 다른 알고리즘 보다 더 우수하게 나타나고 있다. 생성된 예제 데이타의 특성에 따라 실험 결과가 다소 다르게 나오는 부분은 있으나 이동 객체가 전체 영역에 넓게 퍼져 있는 경우에 최소지지도 20%는 후보 시공간 이동 패턴을 많이 생성하지 못하기 때문에 STMPE와 STPMine2 간에는 큰 차이가 없는 것을 보여주고 있다.

5. 결 론

본 논문에서 대용량의 시공간 데이타 집

합으로부터 이동 객체의 이동 패턴을 효율적으로 추출하기 위한 STMPE 알고리즘을 제안하였다. STMPE 알고리즘은 이동 객체의 이력 정보를 분석하여 시공간 정보를 나타내는 일반화된 이동 패턴을 추출할 수 있으며, 이를 위해서 시공간 데이타 일반화 기능, 단기 패턴 생성 기능, 패턴 트리 생성 기능 등을 지원한다.

시공간 데이타 일반화 기능은 이동 객체의 위치 및 시간 데이타를 사용자의 정의에 따라 자동 생성되는 일반화 값으로 변환한다. 이로 인하여 데이타를 표현하는데 필요한 공간의 크기를 줄일 수 있으며, 시공간의 특성을 부여한 이동 패턴을 추출할 수 있다. 단기 이동 패턴 추출 기능은 일반화된 시간 영역에서 연속적인 두 개의 시간 영역에 대한 이동 패턴, 즉 단기 이동 패턴을 추출한다. 추출된 단기 이동 패턴은 이동 패턴 트리를 구성하는 최소 이동 패턴이다. 또한 각 시간 영역 별로 단기 이동 패턴 추출 연산을 수행하기 때문에 대용량의 시공간 데이타베이스를 시간 영역 별로 나누어 입력할 수 있다. 이동 패턴 트리 생성 기능은 단기 이동 패턴을 시간의 순서에 따라 트리로 구성한다. 따라서 다양한 질의에 대하여 빠른 검색을 수행할 수 있게 한다. 이렇게 구축된 이동 패턴 트리는 시공간 이동 패턴이 필요한 실세계의 다양한 응용 서비스에서 활용될 수 있다.

본 논문에서 제안한 STMPE 알고리즘은 시공간 데이타를 일반화시킴으로서 대용량 시공간 데이타 저장으로 인한 메모리 사용량을 최소화하며, 데이타베이스 스캔시 단기 이동 패턴을 작성하여 유지함으로서 데이타베이스 스캔 횟수를 최소화하여 수행 시간을 단축할 수 있다. 또한 STMPE 알고리즘은 수행 시간이 짧고 소요 메모리의 양이 적기 때문에 최소지지도가 낮은 시공간 이동 패턴을 추출하는 경우에도 효율적으로 적용이 가능하다. 그래서 대용량의 시공간 데이타에서 최소지지도가 높거나 낮은 모든 경우에 대하여 수행 시간 및 메모리 공간의 비용이 비슷하게 소요되므로 효율적인 시공간 이동 패턴을 추출할 수 있다.

본 논문에서 제안한 STMPE 알고리즘은 시공간 이동 패턴 알고리즘에 대해 이동 객체 수, 최소지지도, 시간 분할 횟수 등의 기준 값의 변화에 따른 성능 평가를 시공간 이동 패턴 추출에 따른 수행 시간과 소요 메모리의 양에 대해서 수행하였고 성능 실험을 통하여 성능을 평가한 결과는 다음과 같다. 최소지지도가 낮아짐에 따라 수행 시간과 메모리의 양에 대해 2배에서 10배 이상의 성능 향상을 보였고, 이동 객체 수가 증가함에 따라 2배 이상의 성능 향상을 보였으며, 시간 분할 횟수의 증가에 따라 10배 이상의 성능 향상을 보였다. 결과적으로 STMPE 알고리즘이 모든 부분에서 시간 정보를 갖는 다른 시공간 이동 패턴 추출 알고리즘보다 최소지지도가 낮아질수록, 이동 객체의 수가 증가할수록, 시간 분할 횟수가 많아질수록 더 뛰어난 성능을 보였다.

STMPE 알고리즘을 기반으로 한 향후 연구과제로는 그리드 기반으로 시공간 이동 패턴을 추출하는데 있어서 대상이 되는 공간 영역을 공간 특성 및 의미를 활용하여 공간 영역을 표현하는 셀을 다중 분할하여 일반화하는 부분과 분석 대상의 이력 데이타에서 불필요한 데이타를 제거하여 효율적인 이동 패턴이 될 수 있도록 여러 가지 제약조건을 정의하여 데이타 마이닝의 전처리 과정에서 필요 없는 데이타를 필터링하는 부분 등이 있다.

참고문헌

1. 이준욱, 이용준, 류근호, "시간 데이타마이닝 프레임워크," 한국정보과학회 논문지D,

Vol.9-D, No.3, 2002, pp.365-389.
2. Han, J., Dong, G., and Yin, Y., "Efficient Mining of Partial Periodic Patterns in Time Series Database," Proceedings of International Conference on Data Engineering, 1999, pp.106-115.
3. Mamoulis, N., Cao, H., Kollios, G., Hadjieleftheriou, M., Tao, Y., and Cheung, D. "Mining, Indexing, and Querying Historical Spatiotemporal Data," Proceedings of 10th Knowledge Discovery and Data Mining (KDD), 2004, pp. 236-245.
4. Tsoukatos, E., and Gunopoulos, D., "Efficient Mining of Spatio-Temporal Patterns," ACM Symposium on Spatial and Temproral Databases, 2001, pp.214-223.
5. Agrawal, R., and Srikant, R., "Fast Algorithms for Mining Association Rules." Proceedings of VLDB, 1994, pp. 487-499.
6. Berger, G., and Tuzhilin, A., "Discovering Unexpected Patterns in Temporal Data using Temporal Logic," Temporal Databases Research and Practice, Springer-Verlag, 1998.
7. Mohammed, J., Zaki, "SPADE: An Efficient Algorithm for Mining Frequent Sequences," Machine Learning, 2000, pp.1-31.
8. Yavas, G., Katsaros, D., Ulusoy, O., and Manolopoulos, Y., "A data mining approach for location prediction in mobile environments," Data & Knowledge Engineering(DKE), 2005, pp.121-146.

박지웅
1992년 배재대학교 전자계산학과 졸업(이학사)
1994년 건국대학교 대학원 컴퓨터공학과 졸업(공학석사)
2006년 건국대학교 대학원 컴퓨터공학과 졸업(공학박사)
2001년~현재 경기공업대학 컴퓨터정보시스템과 조교수
관심분야 : 시공간데이타베이스, GIS, LBS

김동오
2000년 건국대학교 컴퓨터공학과 졸업(공학사)
2002년 건국대학교 대학원 컴퓨터공학과 졸업(공학석사)
2006년 건국대학교 대학원 컴퓨터공학과 졸업(공학박사)
2006년 건국대학교 강의 교수
관심분야 : 유비쿼터스 GIS, 센서 DBMS, MODB, LBS, GML

홍동숙
1999년 건국대학교 컴퓨터공학과 졸업(공학사)
2001년 건국대학교 대학원 컴퓨터공학과 졸업(공학석사)
2000년~2003년 쌍용정보통신 모바일/GIS 기술팀
2003년~현재 건국대학교 대학원 컴퓨터공학과 박사과정
관심분야 : LBS, 이동체 데이터베이스, 유비쿼터스 컴퓨팅

한기준
1979년 서울대학교 수학교육학과 졸업(이학사)
1981년 한국과학기술원(KAIST) 전산학과 졸업 (공학석사)
1985년 한국과학기술원(KAIST) 전산학과 졸업 (공학박사)
1985년~현재 건국대학교 컴퓨터공학부 교수
1990년~1991년 Stanford 대학 전산학과 visiting scholar
2000년~2002년 한국정보과학회 데이타베이스연구회 운영위원장
2004년~ 2006년 한국공간정보시스템학회 회장
2004년~현재 한국정보시스템감리사협회 회장
관심분야 : 공간데이타베이스, GIS, LBS, 텔레매틱스