DOI: 10.3745/KIPSTD.2009.16-D.1.113

이동 시퀀스의 빈발도를 이용한 최적 이동 패턴 탐사 기법

이 연 식[†] · 고 현[††]

요 약

기존의 패턴 탐사 기법들은 제한된 시간 및 공간영역에서 발생하는 다양한 이동 패턴들 중 단순히 사용자 요구에 적합할 것으로 추정되는 불특정한 빈발 이동 패턴만을 탐사하기 때문에 특정지점들 간의 최적 이동 경로나 정해진 시간 내의 스케줄링 경로 탐색과 같은 복합적인 시간 및 공간 제약 조건을 갖는 최적 이동 패턴을 탐사하는 문제에는 적용하기 어렵다. 이에 본 논문에서는 방대한 이동 객체의 이력 데이터 집합으로부터 복합적인 시간 및 공간 제약을 갖는 최적 이동 패턴을 탐사하는 문제를 보이고, 적용 가능한 위치 기반 서비스로서 최적 이동 경로에 해당하는 패턴을 탐색하기 위한 새로운 패턴 탐사 기법인 STOMP-F를 제안한다. 제안된 기법은 특정한 지점들 사이를 이동한 객체의 패턴들 중 객체가 가장 빈번하게 이동한 경로를 탐색하여 최적 경로로 결정하는 패턴 빈발도를 이용한 탐색 방법으로, 최적 이동 패턴 탐사 과정의 이동 시퀀스 생성 단계에서 객체의 위치 값과 공간영역 간의 위상 관계를 고려하여 이동 객체의 위치 속성에 대한 최하위 수준에서의 공간 일반화를 통해 보다 효율적으로 패턴 탐사를 수행할 수 있다. 제안 방법을 Dijkstra 알고리즘과 A* 알고리즘을 대상으로 실험 평가한 결과 A* 알고리즘의 휴리스틱 가중치에 따라 차이는 있으나 연산 처리 시간을 기준으로 타 알고리즘들 보다 효과적임을 알 수 있다.

키워드 : 이동 객체, 시공간 데이터 마이닝, 패턴 탐사, 최적 경로 탐색

A Method for Optimal Moving Pattern Mining using Frequency of Moving Sequence

Yonsik Lee[†] · Hyun Ko[††]

ABSTRACT

Since the traditional pattern mining methods only probe unspecified moving patterns that seem to satisfy users' requests among diverse patterns within the limited scopes of time and space, they are not applicable to problems involving the mining of optimal moving patterns, which contain complex time and space constraints, such as 1) searching the optimal path between two specific points, and 2) scheduling a path within the specified time. Therefore, in this paper, we illustrate some problems on mining the optimal moving patterns with complex time and space constraints from a vast set of historical data of numerous moving objects, and suggest a new moving pattern mining method that can be used to search patterns of an optimal moving path as a location-based service. The proposed method, which determines the optimal path(most frequently used path) using pattern frequency retrieved from historical data of moving objects between two specific points, can efficiently carry out pattern mining tasks using by space generalization at the minimum level on the moving object's location attribute in consideration of topological relationship between the object's location and spatial scope. Testing the efficiency of this algorithm was done by comparing the operation processing time with Dijkstra algorithm and A* algorithm which are generally used for searching the optimal path. As a result, although there were some differences according to heuristic weight on A* algorithm, it showed that the proposed method is more efficient than the other methods mentioned.

Keywords : Moving Object, Spatio-Temporal Data Mining, Pattern Mining, Optimal Path Search

1. 서 론

최근 다양한 응용 분야에서 활용할 수 있는 새로운 위치 기반 서비스를 개발하고자 이동 객체의 동적인 위치 변화에 따른 특성 분석을 통해 유용한 정보를 추출하기 위한 이동 패턴 탐사에 대한 연구들이 지속적으로 수행되고 있다. 시공간 이동 패턴 탐사는 연속되는 시간영역에서 이동하는 객체의 패턴들 중 위치 변화에 따른 독특한 경향이나 반복적이고 공통적인 의미있는 패턴을 탐사하기 위한 기술이다. 기존의 시공간 이동 패턴 탐사 기법들은 대부분 연속적인 시간의 흐름에 따라 발생하는 객체의 이동 패턴들 중 주기적인 이동 패턴[1, 2]이나 순차적인 이동 패턴[3~6, 11~14]

※본 연구는 한국과학재단 특정기초연구(R01-2007-20989-0) 지원으로 수행되었음.
† 종신회원 : 군산대학교 컴퓨터정보공학과 교수
†† 정 회 원 : 한국항공우주연구원 프로젝트연구원
논문접수 : 2008년 5월 6일
수 정 일 : 1차 2008년 11월 25일
심사완료 : 2008년 12월 11일

의 탐사를 목적으로 하고 있다. 이외에도 이동 객체의 위치 변화에 있어 물리적 또는 시간적으로 유사한 특성을 가진 객체들이 집단화되는 현상으로부터 그룹(group) 패턴을 탐사하기 위한 방법[7,8]과 이동 점 객체들이 군집을 형성하면서 특정한 방향으로 이동하는 움직임으로부터 모션(motion) 패턴을 탐사하기 위한 방법[9,10]이 연구되었다. 하지만 현재까지 연구된 대부분의 탐사 기법들은 순차패턴이나 주기패턴과 같이 반복적이고 공통적으로 발생하는 빈발한 이동 패턴을 탐사하기 위한 방법들로, 새로운 위치 기반 서비스의 개발보다는 단순히 패턴 탐사 성능을 향상시키는데 주목적을 두고 있다. 또한 기존의 탐사 기법들은 시간 및 공간 한정자(qualifier)에 의해 제한된 시간 및 공간 영역에서의 모든 이동 패턴들 중 불특정한 빈발 이동 패턴만을 탐사하기 때문에 시간 및 공간 한정자 이외의 또 다른 시간 및 공간 제약 조건을 가진 시공간 이동 패턴을 탐사하는 문제에 대해서 능동적으로 대처하기 어렵다. 가령, 특정 지점들 간의 최적 이동 경로나 정해진 시간 내에서의 효율적인 스케줄링 경로 탐색과 같이 사용자에 의해 지정된 시공간 범위 내에서 시간 및 공간 제약 조건을 갖는 빈발 패턴을 탐사하는 문제에는 적용하기 어렵다. 따라서 이러한 문제에 대한 근본적인 해결을 통해 보다 효율적인 패턴 탐사를 수행하는 새로운 시공간 이동 패턴 탐사 기법이 필요하다.

이에 본 논문에서는 방대한 시공간 이동 데이터 집합으로부터 다양한 이동 패턴들 중 복합적인 시간 및 공간 제약 조건을 만족하고, 객체가 가장 효율적인 경로를 통해 이동할 수 있도록 최적 경로에 해당하는 최적 이동 패턴을 탐사하기 위한 STOMP-F(Spatio-Temporal Optimal Moving Pattern-Frequency) 알고리즘을 제안한다. 제안된 알고리즘은 특정한 지점들 사이를 이동한 객체의 이력 데이터들 중 객체가 가장 빈번하게 이동한 경로를 탐색하여 최적 경로로 결정하는 패턴 빈발도를 이용한 탐사 방법으로, 최적 이동 패턴 탐사 과정의 전처리 단계(이동 시퀀스 생성 단계)에서 이동 객체의 위치값과 공간영역 간의 위상 관계를 고려하여 이동 객체의 위치 속성에 대한 최하위 수준에서의 공간 일반화를 통해 보다 효율적으로 패턴 탐사를 수행할 수 있다[16].

본 논문의 구성은 2장에서 관련연구로서 기존의 시공간 이동 패턴 탐사 기법에 대해 기술하고, 3장에서는 빈발도 기반의 최적 이동 패턴 탐사 문제에 대한 특성을 정의한다. 4장에서는 최대 빈발 이동 패턴을 추출하기 위한 단계적인 패턴 탐사 과정을 제시하고, STOM 알고리즘과 그의 요소 알고리즘들을 설계 및 구현한다. 5장에서는 실험을 통해 제안된 탐사 기법의 성능을 평가하고, 마지막 6장에서는 결론 및 향후 연구 과제를 제시한다.

2. 관련 연구

시공간 객체의 이동 패턴 탐사는 연속적인 시간의 흐름에 따라 이동하는 시공간 객체의 패턴들 중 객체의 위치 변화에 따른 공통적인 경향이나 의미있는 패턴을 탐사하는 기법으로, 현재까지 연구된 시공간 이동 패턴 탐사 기법들로는 이동 객체가 이동하는 위치 변화의 패턴을 탐사하는 이동 패턴 탐사[1~6, 13, 14]와 이동 객체의 위치 변화에 있어 집단화되는 패턴을 탐사하는 그룹 패턴 탐사[7,8], 이동 객체가 군집을 형성하면서 이동하는 움직임에서 패턴을 탐사하는 모션 패턴 탐사[9, 10] 기법 등이 있다.

Mamoulis[2]는 이동 객체의 이력 데이터 집합으로부터 다양한 이동 객체의 이동 패턴들 중 주기적으로 발생하는 빈발한 이동 패턴을 추출하기 위한 방법으로 STPMine1(Spatial-Temporal Pattern Mine1)과 STPMine2 알고리즘을 제시하였다. Apriori 계열의 알고리즘을 변형한 STPMine1 알고리즘은 빈발 이동 패턴 탐사 시 후보 패턴의 길이가 길어질수록 빈번한 데이터베이스 스캔으로 인해 탐사 수행 성능이 저하되는 문제를 가지고 있다. 이러한 문제로 Mamoulis[2]는 STPMine2에서 데이터베이스의 접근을 최소화하기 위해서 메모리상에 Max_Subpattern 트리를 생성하여 효율적인 패턴 탐사가 이루어지도록 하였다. 또한, STPMine2와 유사한 방법으로 Kim[1, 15]도 이동 패턴 트리를 생성하여 대용량의 시공간 데이터 집합으로부터 이동 객체의 이동 패턴을 효율적으로 추출하기 위한 STMPE(Spatio-Temporal Moving Pattern Extracting) 알고리즘을 제시하였다. STMPE는 이동 객체의 이력 정보를 분석하여 시간 및 공간 정보를 모두 포함하는 일반화된 이동 패턴을 추출하는 알고리즘으로, 시공간 데이터 일반화 기능과 이동 패턴 트리 생성 기능을 지원함으로써 패턴 탐사 시 데이터를 표현하는데 필요한 저장 공간의 크기를 감소시키고 데이터베이스의 스캔 횟수를 감소시켜 탐사 수행 시간을 최소화하도록 하였다. 이러한 연구들은 패턴 탐사를 통해 추출되는 지식들이 실세계의 다양한 분야에서 제공되도록 위치 기반 서비스 개발을 목적으로 수행되었기 보다는 패턴 탐사 수행 성능의 향상을 목적으로 연구되었기 때문에 특정 응용 분야에 적용가능한 새로운 위치 기반 서비스에 대한 개발이 필요하다.

하지만 일부에서는 이동 객체의 빈발한 이동 패턴을 추출하여 특정 분야에 적합한 위치 기반 서비스로 활용하기 위한 연구들이 시도되었다. Yavas[6]는 PCS(Personal Communication System) 네트워크 안에서 사용자의 이동성 예측(mobility prediction)을 위해서 이동성 패턴을 추출하는 UMP(User Mobility Pattern) 마이닝 기법을 제안하였다. 또한 한선영[14]은 다양한 이동 객체들의 공통적인 이동 패턴을 분석하여 그 패턴에 따라 미술관이나 박물관에서의 전시물 배치, 쇼핑몰에서의 새로운 제품 판촉이나 상품 배치 등과 같은 위치 기반 서비스를 제공하기 위한 패턴 탐사 기법인 Apriori-MSP 알고리즘을 제시하였다. 이러한 연구들처럼 시공간 객체의 주기적이거나 반복적인 빈발 이동 패턴의 추출을 통해 실세계의 보다 다양한 분야에 적용할 수 있는 새로운 위치 기반 서비스의 개발이 아직은 부족한 실정임으로 이에 대한 연구가 지속적으로 필요하다.

한편 지금까지 고찰한 기존의 이동 탐사 기법들은 시간 및 공간 한정자(qualifier)에 의해 제한된 시간 및 공간 영역에서 발생한 모든 이동 패턴들 중 불특정한 빈발 이동 패턴

들만을 탐사하기 때문에 시간 및 공간 한정자 이외의 또 다른 시간 및 공간 제약 조건을 가지는 의미있는 이동 패턴을 탐사하기에는 문제가 있다.

3. 빈발도 기반의 최적 이동 패턴 탐사 정의

최적 이동 패턴 탐사는 방대한 이동 객체의 이력 데이터 집합으로부터 특정 지점들 간을 이동하는 객체들의 다양한 패턴들 중 가장 최적의 비용을 소요하는 이동 패턴의 추출을 목적으로 한다. 최적 이동 패턴을 탐사하는 문제는 기존의 패턴 탐사 문제와는 다르게 탐색하고자 하는 대상 패턴을 추출하기 위해서 시간적인 제약이나 공간적인 제약, 혹은 두 제약 모두를 수반하게 된다. 이러한 제약은 기존의 패턴 탐사 문제에서 단순한 탐사 범위의 시간적인 혹은 공간적인 한정과는 다르다. 가령, 패턴 탐사 대상이 되는 이동 객체의 이력 데이터에 대해 기존의 패턴 탐사에서의 공간 한정은 특정한 공간 내에서 발생하는 모든 패턴이 대상이 될 수 있지만, 최적 이동 패턴에서의 공간 제약은 사용자의 요구에 따라 제한될 수 있는 특별한 위치점들이나 공간 영역들 사이에서 발생할 수 있는 모든 이동 데이터들이 대상이 될 수 있기 때문이다. 현재까지 연구된 패턴 탐사 기법들은 이러한 복합적인 시간 및 공간 제약을 가지는 이동 패턴들을 추출하기 어렵다. 이는 기존의 탐사 기법들이 시간 및 공간 한정자에 의해 제한된 범위의 시간 및 공간 영역에서 반복적이거나 주기적 혹은 공통적으로 발생하는 빈발 패턴 탐사를 주목적으로 함으로써 특정한 제약을 가진 패턴을 탐사하는 문제에 쉽게 대처할 수 없기 때문이다. 따라서 이러한 문제를 해결하여 보다 효율적으로 패턴 탐사를 수행하기 위한 가진 새로운 시공간 이동 패턴 탐사 기법이 필요하다.

빈발도를 기반으로한 최적 이동 패턴 탐사에서 최적 이동 패턴이란 객체의 다양한 이동 패턴들 중 최대지지도를 가지는 가장 빈발도가 높은 이동 패턴으로, 빈발도 기반의 패턴 탐사는 특정한 지점들 간을 이동한 객체들의 패턴들 중 가장 빈번하게 발생하는 이동 패턴을 탐사하기 위한 방법이다. 제안된 방법의 기본 개념은 특정한 지점 간을 이동하는 다양한 패턴들 중 가장 빈번하게 발생하는 패턴이 가장 최적의 비용을 소요할 것이라는 가정을 기반으로 한다. 시간 및 공간 제약 조건을 만족하는 이동 패턴들의 빈발도를 이용하여 최적 패턴을 탐색하기 위한 정의는 다음과 같다.

[정의 1] 이동 객체 데이터베이스 *MD*, 사용자가 지정한 특정지점들 *S*와 *F*(공간제약), 시간기간 T_1과 T_2, 사용자가 지정한 최소지지도 *min_sup*, 최소지지도를 만족하는 *S*에서 *F*까지의 모든 빈발 이동 시퀀스를 추출하여 각 부분 시퀀스들 중 지지도가 가장 높은 부분 시퀀스들의 집합을 탐색하는 것이다. 단, 부분 시퀀스는 *S*에서 *F*까지를 도달하는 하나의 시퀀스에 포함되어야 한다.

4. 빈발도 기반의 최적 이동 패턴 탐사 기법

본 장에서는 3장에서 제시한 최적 이동 패턴 탐사 문제에 대한 정의를 기반으로 빈발도가 가장 높은 이동 패턴을 탐사하는 과정과 탐사 알고리즘을 제시한다.

4.1 최적 이동 패턴 탐사 과정

이동 패턴 탐사 과정은 이동 객체의 이력 데이터로부터 이동 시퀀스를 생성하는 전처리 단계와 이동 패턴 탐사 단계로 구분된다.

4.1.1 전처리 단계 : 이동 시퀀스 생성 단계

이동 시퀀스는 이동 객체의 이력 데이터를 각 이동 객체별로 시간 순차에 따라 나열한 한 쌍의 공간 및 시간 속성값으로 구성된 순차 리스트이다. 이러한 이동 시퀀스의 생성은 <표 1>과 같이 이동 객체 데이터베이스를 각 이동 객체별로 유효시간을 기준으로 정렬한 후 시퀀스를 구성하는 위치들 간에 최대 시간 간격에 대한 제약 조건을 두어 이를 만족해야만 의미있는 이동 시퀀스로 생성될 수 있도록 한다. <표 1>에서 OID는 유일한 객체 식별자이고, (X,Y)는 2차원 공간상에서 이산적으로 샘플링된 객체의 위치 좌표 값, VT는 위치 (X,Y)에서 샘플링된 시간 값이다.

이동 객체별 순차리스트는 사건(위치 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 간격을 만족해야만 하나의 트랜잭션, 즉 시퀀스로 생성될 수 있다. 시간 제약 조건이란 시퀀스 내에서의 연속적인 이동으로 특정 위치에서 인접한 다른 위치로 이동한 경우, 인접한 위치로 이동이 발생한 시간 $t_j - t_{j-1}$을 의미한다. 최대 시간 간격은 *max_gap*으로 $t_j - t_{j-1} \le max_gap$이고, $2 \le j \le k$이다. 즉, 이동 객체의 공간 속성에 대한 샘플링 시간을 검사하여 특정 위치에 머문 시간이 최대 시간 간격 *max_gap*을 초과하면 초과 이전까지의 이동 객체의 순차리스트에 대한 이동 시퀀스와 초과 이후의 이동 시퀀스로 분리한다. 가령, [표 1]에서 객체 4의 이동 시퀀스는 각 위치점 간의 이동 시간 간격을 고려하지 않았을 때 (95,99)→(139,108)→(157,90)→(157,53)→(163,48)→(191,81)→(226,66)로 표현될 수 있으나, *max_gap*이 최대 1시간이라면 (157,53)에서 멈추었다가 다시 출발한 시간이 1시간 이상이기 때문에 <(95,99)(139,108)(157,90) (157,53)>과 <(163,48)(191,81)(226,66)>, 두 개의 시퀀스로 분리되어야 한다. 마찬가지로 객체 7의 이동 시퀀스도 (78,90)→(100,70)→(116,87)→(139,72)→(157,90)→(191,81)→(226,66)로 표현되나 <(78,90)(100,70)(116,87)(139,72)>과 <(157,90)(191,81)(226,66)>로 분리된다. 다음 <표 2>는 <표 1>의 이력 데이터를 시간 간격(*max_gap*=1시간)을 고려하여 이동 시퀀스로 생성한 것이고, <표 3>은 <표 2> 이동 시퀀스의 공간 속성(X,Y)을 노드 식별자 N={N1,N2,…,Nn}로 대치하여 생성한 이동 시퀀스 집합인 트랜잭션 데이터베이스이다.

〈표 1〉 이동 객체의 이력 데이터 예

OID	X	Y	VT	OID	X	Y	VT
1	84	166	2005/11/20/12/10/08	6	84	166	2005/11/20/12/10/24
	108	146	2005/11/20/12/21/19		108	146	2005/11/20/12/27/15
	121	133	2005/11/20/12/29/34		88	125	2005/11/20/12/36/34
	95	99	2005/11/20/12/43/36		95	99	2005/11/20/12/50/45
	139	108	2005/11/20/12/56/16		139	108	2005/11/20/13/02/12
	157	90	2005/11/20/13/04/25		157	90	2005/11/20/13/10/54
	191	81	2005/11/20/13/11/16		191	81	2005/11/20/13/22/24
	226	66	2005/11/20/13/15/16		226	66	2005/11/20/13/31/14
2	88	125	2005/11/20/12/02/55	7	78	90	2005/11/20/12/37/43
	95	99	2005/11/20/12/09/41		100	70	2005/11/20/12/45/39
	116	87	2005/11/20/12/15/15		116	87	2005/11/20/12/53/26
	139	72	2005/11/20/12/21/13		139	72	2005/11/20/12/59/34
	157	90	2005/11/20/12/26/15		157	90	2005/11/20/13/16/14
	191	81	2005/11/20/12/33/34		191	81	2005/11/20/13/27/12
	226	66	2005/11/20/12/41/45		226	66	2005/11/20/13/36/32
3	84	166	2005/11/20/12/33/35	8	116	87	2005/11/20/12/14/35
	108	146	2005/11/20/12/54/14		100	70	2005/11/20/12/23/54
	121	133	2005/11/20/13/15/11		119	52	2005/11/20/12/33/27
	139	108	2005/11/20/13/29/12		134	51	2005/11/20/12/39/19
	157	90	2005/11/20/13/43/36		163	48	2005/11/20/12/48/35
	191	81	2005/11/20/13/52/29		191	81	2005/11/20/13/00/26
	226	66	2005/11/20/13/01/19		226	66	2005/11/20/13/07/07
4	95	99	2005/11/20/12/10/05	9	150	169	2005/11/20/12/31/53
	139	108	2005/11/20/12/20/54		156	160	2005/11/20/12/34/24
	157	90	2005/11/20/12/28/29		148	134	2005/11/20/12/42/34
	157	53	2005/11/20/12/39/38		173	108	2005/11/20/12/49/25
	163	48	2005/11/20/13/00/27		199	108	2005/11/20/12/56/43
	191	81	2005/11/20/13/12/06		191	81	2005/11/20/13/05/16
	226	66	2005/11/20/13/20/46		226	66	2005/11/20/13/12/15
5	121	133	2005/11/20/12/15/45	10	195	166	2005/11/20/12/15/04
	95	99	2005/11/20/12/22/54		182	151	2005/11/20/12/20/43
	139	108	2005/11/20/12/34/07		173	108	2005/11/20/12/29/48
	157	90	2005/11/20/12/45/46		199	108	2005/11/20/12/34/23
	191	81	2005/11/20/12/55/15		191	81	2005/11/20/13/20/26
	226	66	2005/11/20/13/04/33		226	66	2005/11/20/13/26/00

〈표 2〉 단위 이동 시퀀스 추출 예

[이동 객체의 이력 데이터]

OID	X	Y	VT
...	...	...	...
4	95	99	2005/11/20/17/10/05
	139	108	2005/11/20/17/20/54
	157	90	2005/11/20/17/28/29
	157	53	2005/11/20/17/39/38
	163	48	2005/11/20/18/51/27
	191	81	2005/11/20/19/12/06
	226	66	2005/11/20/19/20/46
...	...	...	...
7	78	90	2005/11/20/18/37/43
	100	70	2005/11/20/18/45/39
	116	87	2005/11/20/18/53/26
	139	72	2005/11/20/18/59/34
	157	90	2005/11/20/21/16/14
	191	81	2005/11/20/21/27/12
	226	66	2005/11/20/21/36/32
...	...	...	...

divide

divide

OID	USID	X	Y	VT
...	...	...	...	...
4	S4	95	99	2005/11/20/17/10/05
	S4	139	108	2005/11/20/17/20/54
	S4	157	90	2005/11/20/17/28/29
	S4	157	53	2005/11/20/17/39/38
	S5	163	48	2005/11/20/18/51/27
	S5	191	81	2005/11/20/19/12/06
	S5	226	66	2005/11/20/19/20/46
...	...	...	...	...
7	S8	78	90	2005/11/20/18/37/43
	S8	100	70	2005/11/20/18/45/39
	S8	116	87	2005/11/20/18/53/26
	S8	139	72	2005/11/20/18/59/34
	S9	157	90	2005/11/20/21/16/14
	S9	191	81	2005/11/20/21/27/12
	S9	226	66	2005/11/20/21/36/32
...	...	...	...	...

〈표 3〉 트랜잭션 데이터베이스

OID	Sequence
MO1	<N1 N3 N15 N17 N14 N19 N20 N21>
MO2	<N16 N17 N22 N23 N19 N20 N21>
MO3	<N1 N3 N15 N14 N19 N20 N21>
MO4	<N17 N14 N19 N24> <N25 N20 N21>
MO5	<N15 N17 N14 N19 N20 N21>
MO6	<N1 N3 N16 N17 N14 N19 N20 N21>
MO7	<N30 N29 N22 N23> <N19 N20 N21>
MO8	<N22 N29 N28 N26 N25 N20 N21>
MO9	<N6 N7 N13 N12 N10 N20 N21>
MO10	<N3 N15 N14 N19> <N20 N21>

4.1.2 최적 이동 패턴 탐사 단계

빈발도 기반의 최적 이동 패턴 탐사는 전처리 과정을 통해 생성된 이동 시퀀스 집합으로부터 최대지지도를 가지는 가장 빈발한 이동 시퀀스를 탐사하는 방법으로, 후보 시퀀스 집합 생성 단계, 최대 빈발 2-시퀀스 추출 단계, 최적 이동 패턴 추출 단계의 과정을 거쳐 최대지지도를 가지는 빈발 이동 시퀀스를 추출한다. 이 때 지지도는 연관규칙 탐사 방법에서의 지지도 개념과 같이, 2-시퀀스를 포함하는 트랜잭션 수의 전체 트랜잭션 수에 대한 비율을 나타낸다.

[단계 1] 후보 시퀀스 집합 생성 단계

(1) 전체 이동 시퀀스 집합 MDB로부터 각 시퀀스를 구성하는 항목들 중 첫 번째 항목이 S이고 마지막 항목이 D인 후보 시퀀스 집합 C_1를 추출한다.

(2) 각 시퀀스에서 S와 다음 항목 S′으로 구성된 2-시퀀스 집합 F_1를 추출한다.

[단계 2] 최대 빈발 2-시퀀스 추출 단계

(3) 전체 이동 시퀀스 집합을 읽어들여 [단계 1]에서 추출한 각 2-시퀀스에 대한 지지도를 계산하여 최대지지도를 갖는 빈발 2-시퀀스를 추출한다.

(4-i) 빈발 2-시퀀스의 두 번째 항목 S′이 마지막 항목 D와 같지 않으면 [단계 1]을 반복하고 같으면 [단계 3]을 수행한다.

(4-ii) 만약 빈발 2-시퀀스가 2이상일 경우 각각의 S′과 마지막 항목 D와 같지 않으면 [단계 1]을 반복하고 같을 경우 [단계 3]을 수행한다.

[단계 3] 최적 이동 패턴 추출 단계

(5) [단계 1]과 [단계 2]의 과정을 거쳐 생성된 최대 빈발 2-시퀀스들을 순차적으로 연결하여 최적 이동 패턴을 추출한다.

(그림 1)은 출발지점 S가 N1이고 도착지점 D가 N21이라고 할 때, <표 3>의 이동 시퀀스 집합으로부터 N1에서 D21까지의 최대지지도를 가진 빈발 이동 시퀀스를 탐사하는 최적 이동 패턴 탐사 예이다. (그림 1)의 PASS 1에서는 MDB로부터 N1과 N21항목을 포함하는 후보 시퀀스 집합 C_1을 추출한다. 그 다음 C_1을 구성하는 각 시퀀스의 1번째 항목과 2번째 항목을 추출하여 부분 2-시퀀스를 생성하고 지지도를 계산하여 2-시퀀스 집합인 F_1을 구성한다. F_1의 시퀀스는 <N1 N3>로 단 한개만 생성되어 최대 빈발 2-시퀀스로 결정한다.

PASS 2에서는 PASS 1에서 추출한 빈발 2-시퀀스인 <N1 N3>에서 2번째 항목 N3을 새로운 출발지점 S′으로 지정하고, MDB로부터 N3과 N21 항목을 포함하는 후보 시퀀스 집합 C_2를 추출한다. PASS 1과 같이 C_2를 구성하는 각 시퀀스의 1번째와 2번째 항목을 추출하여 부분 2-시퀀스를 생성하고 각 시퀀스의 지지도를 계산하여 F_2를 구성한다. F_2의 시퀀스들로는 <N3 N15>와 <N3 N16>이 생성되는데, 지지도가 각각 0.3과 0.1이기 때문에 지지도가 더 큰 <N3 N15>를 최대 빈발 2-시퀀스로 결정한다.

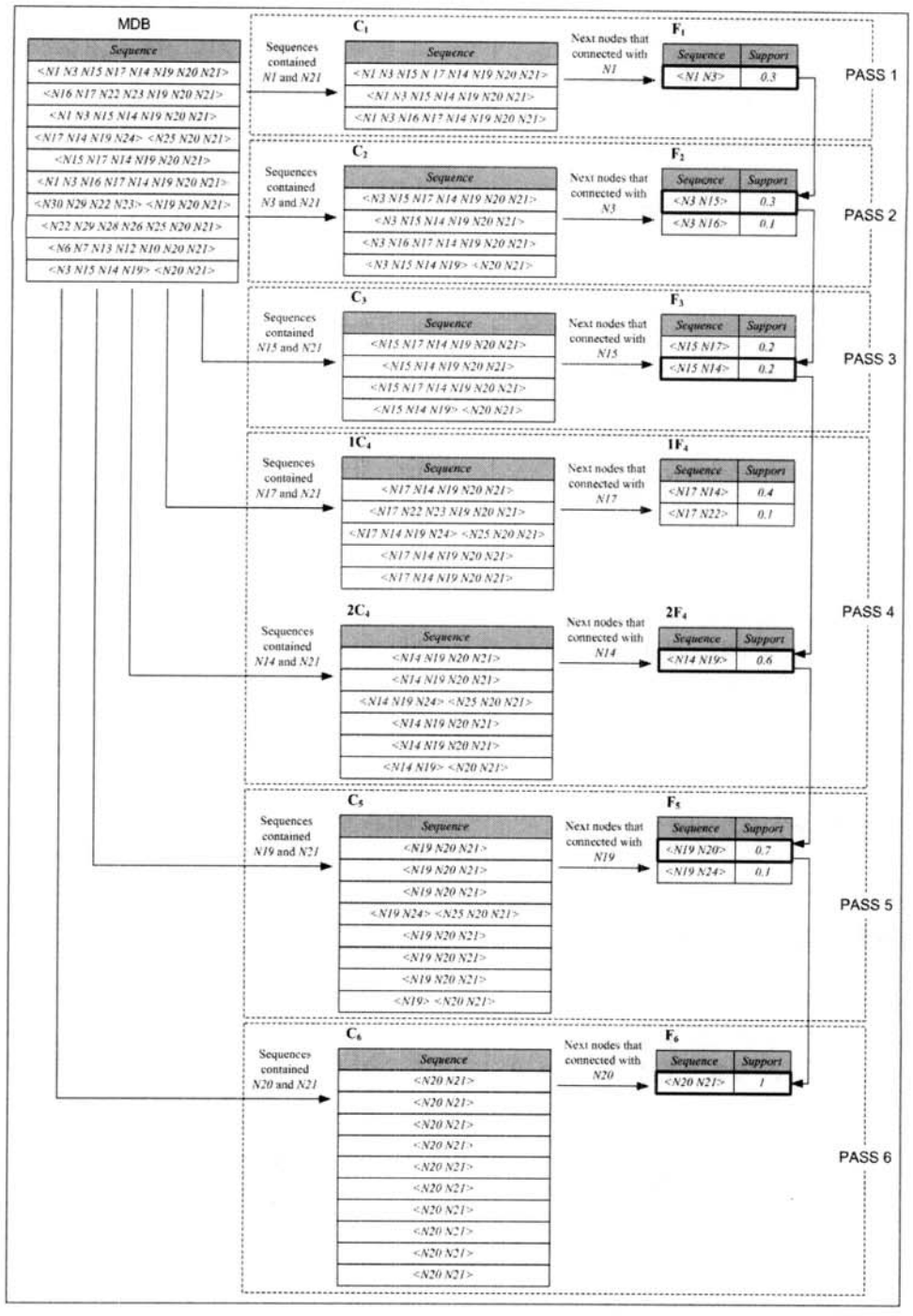
한편, PASS 3에서는 N15와 N21 항목을 가지는 후보 시퀀스 집합 C_3을 구성하고, C_3의 각 시퀀스들로부터 추출된 부분 2-시퀀스 <N15 N17>과 <N15 N14>에 대한 지지도를 계산하여 F_3을 구성한다. 최대 빈발 2-시퀀스의 추출은 2-시퀀스 집합 F_i의 각 시퀀스에 대한 지지도를 비교하여 추출하게 되는데, F_3의 두 부분 2-시퀀스는 동일한 지지도를 가짐으로써 하나의 최대 빈발 2-시퀀스를 추출할 수 없다. 이러한 경우 두 시퀀스 각각을 대상으로 다음의 탐사 패스를 수행하여 추출되는 모든 부분 2-시퀀스들 중 가장 큰 지지도를 가지는 최대 빈발 2-시퀀스를 탐사한다.

PASS 4에서는 PASS 3의 두 부분 2-시퀀스 <N15 N17>과 <N15 N14>를 기반으로 후보 시퀀스 집합 $1C_4$와 $2C_4$를 구성하고, 다시 $1C_4$와 $2C_4$로부터 부분 2-시퀀스들을 생성한 후 각각의 지지도를 계산하여 $1F_4$와 $2F_4$를 추출한다. PASS 4의 최대 빈발 2-시퀀스는 $1F_4$와 $2F_4$의 모든 2-시퀀스들에 대한 지지도 중 가장 높은 지지도를 가지는 <N14 N19>가 결정된다. PASS 3 단계의 최대 빈발 2-시퀀스로는 PASS 4의 최대 빈발 2-시퀀스가 N14를 포함함으로 <N15 N14> 시퀀스가 최대 빈발 2-시퀀스로 결정된다.

이러한 탐사 과정을 반복적으로 수행하여 도착지점인 N21을 포함하는 최대 빈발 2-시퀀스가 결정되면 탐사 과정을 마친다. 최종적으로 각 PASS에서 결정된 최대 빈발 2-시퀀스를 조합하여 하나의 최적 이동 패턴을 생성한다. (그림 1)에서는 <N1 N3 N15 N14 N19 N20 N21> 시퀀스가 가장 최적의 이동 패턴으로 결정된다.

4.2 최적 이동 패턴 탐사 알고리즘

최적 이동 패턴 탐사 알고리즘인 STOMP-F는 SeqExtractor 함수를 통해 이동 시퀀스를 생성한 후 다시 OptPathExtractor 함수를 통해 최대 빈발 시퀀스를 추출하여 최적 경로를 결정한다. Contains[16] 함수는 이동 객체의 위치 속성을 실세



(그림 1) 빈발도 기반의 최적 이동 패턴 탐사 예

계의 공간지역으로 일반화하기 위한 함수이다. (그림 2)의 STOMP-F 알고리즘은 먼저, limitedDataSet 함수를 호출하여 이동 객체 데이터베이스에서 공간 및 시간 제약 조건을 만족하는 이동 객체의 이력 데이터 집합을 추출한다. limitedDataSet 함수는 이동 객체 데이터베이스로부터 공간 및 시간 한정자 C_s와 C_t 를 이용하여 C_s 공간영역에서 C_t 시간기간 동안 이동한 객체의 이력 데이터를 추출한 후 다시 공간 제약 조건 S와 F를 이용하여 공간영역의 특정지점들 간을 이동한 객체의 이력 데이터를 추출하기 위한 함수이다. 추출된 이력 데이터 집합은 SeqExtractor 함수를 호출하여 각 이동 객체의 연속된 이동 이력에 대해 시간 간격 조건 *max_gap*을 적용하여 이동 시퀀스 데이터를 생성한다. SeqExtractor 함수는 이산적으로 샘플링된 연속적인 이동 객체의 이력 데이터를 이용하여 패턴 탐사를 수행하기에는 부적합하기 때문에 시간 속성에 대한 시간 간격 제약 조건 *max-gap*을 적용함으로써 이력 데이터를 분할하여 이동 시퀀스를 생성하는 함수이다. 생성된 이동 시퀀스 데이터는 공간 일반화 연산 함수인 Contains를 통해 이동 시퀀스를 구성하는 각 항목들의 공간속성을 일반화한다. 일반화된 이동 시퀀스 집합은 최적 이동 패턴을 추출하기 위한 OptPathExtractor 함수를 통해 최대 빈

```
Input : D(Database), C_s(Spatial Constraint), C_t(Temporal Constraint),
        S(Start Node), F(Final Node), max_gap(Time Interval Constraint),
        Ls_k(Spatial Level)
Output : OptPath

Procedure STOMP-F(D, C_s, C_t, S, F, max_gap, Ls_k)

Begin
   PrevA = null;

   //Set of History Data satisfied Constraint of Spatial and Temporal
   D_rt = limitedDataSet(D, C_s, C_t, S, F);

   // Extracting Set of Moving Sequence from D_rt
   D_seq = SeqExtractor(D_rt, max_gap);

   // Generalizing Region to Spatial Property of Moving Sequence in D_seq
   D_Rgen = Contains(D_seq, PrevA, Ls_k);

   //Moving Pattern Minging
   OptPath = OptPathExtractor(D_Rgen, S, F);

   Return OptPath;

End
```

(그림 2) STOMP-F 알고리즘

```
Input : D_Rgen(Generalized Moving Sequence), S(Start Node), F (Final Node)
Output : OptPath

Procedure OptPathExtractor (D_Rgen, S, F)

Begin
      PrevA = null;
      OptPath = null;
      nextN = S;

      While (nextN ≠ F) {
            OptPath add nextN;
            nextN = Freq_Link(nextN, F, D_Rgen);
      }

      OptPath add F;

      Return OptPath;

End
```

(그림 3) OptPathExtractor 알고리즘

발 패턴인 최적 이동 패턴을 추출한다.

4.2.1 최적 경로 추출 알고리즘 : OptPathExtractor

OptPathExtractor 함수는 최적 이동 패턴을 구성하는 각각의 단위 패턴을 찾기 위해 최초의 출발점 S를 OptPath 변수에 저장하고 S와 연결된 노드를 탐색하기 위해 Freq_Link 함수를 호출한다.

4.2.2 단위 최적 패턴 추출 알고리즘 : Freq_Link

단위 최적 패턴을 추출하기 위한 Freq_Link 함수는 OptPathExtractor 함수에서 전달받은 전체 이동 시퀀스 집합으로부터 출발지점 S와 도착지점 F를 포함하는 부분 이

```
Input : currN(Current Node), F(Final Node), D_Rgen(Generalized Moving Sequence)
Output : Node

Procedure Freq_Link(currN, F, D_Rgen)

Begin
    subD= Search Sequence that contained Node currN and Node F in D_Rgen;
    For each moving_sequence S ∈ subD
      c = subset(currN, S); //Link from currN to nextN that connected with currN
      If c ∉ C_s Then     //Set of Links between the decided Node and optional Node
          C_s add c;
      End If
      c.count++;
    End For

    c_max_sup = find 2-sequence with maximum support from C_s;
    nextN = another Node that connected with currN in c_max_sup;

    Return nextN;
End
```

(그림 4) Freq_Link 알고리즘

동 시퀀스 집합을 추출하여 최대 빈발 이동 패턴을 탐사하게 된다. 이동 패턴을 탐사하는 과정은 최초 S로부터 시작하는 1-최대 빈발 패턴을 탐색하고 그 패턴과 연결된 S가 아닌 반대편 노드를 기준으로 하여 다시 F를 포함하는 모든 부분 이동 시퀀스 집합을 추출하여 다시 1-최대 빈발 패턴을 추출한다. 이러한 과정은 F에 도달하는 1-최대 빈발 패턴을 탐사할 때까지 반복 수행한다.

5. 실험 및 성능 평가

제안된 최적 이동 패턴 추출 알고리즘 STOMP-F에 대한 평가는 최적 경로 탐색을 위해 일반적으로 가장 많이 사용되는 Dijkstra와 A* 알고리즘들을 대상으로 실험을 통해 성능을 비교하였다. 비교 실험을 위한 Dijkstra 알고리즘은 특정지점 간의 최적 경로를 탐색하고자 할 때, 출발점 S에 인접한 노드를 검사하여 최소 가중치(거리, 시간, etc)를 소요하는 노드 u를 찾고 다시 노드 u에서 최소 가중치 v를 선택하는 방식으로 구현하였다. 또한 A* 알고리즘은 특정지점 간의 최적 경로를 탐색하고자 할 때, 휴리스틱 가중치(거리, 시간, etc)에 따라 순서대로 인접노드를 탐색하는 방식으로 구현하였다. 실험 조건은 모두 동일하게 각 100, 200, 300, 400, 500번의 경로 탐색을 통해 평균적인 데이터를 산출하여 결과값에 반영하였다. 실험을 위해 Dijkstra 알고리즘과 A* 알고리즘의 가중치로서 거리, 시간을 기준으로 설정하였고, STOMP_F 알고리즘은 빈발도만을 기준으로 측정하였다. 5장에서의 실험 결과는 가중치로 거리를 기준으로 했을 때의 결과이다. 또한, A* 알고리즘의 경우는 휴리스틱 가중치를 한 범위만 지정할 수 없기 때문에 10~20Km 구간, 20~30Km구간으로 나누어 실험하였다. 최적 경로 탐색 알고리즘들의 성능을 평가하기 위한 실험 기준은 평균 탐색 연산 시간과 최적 경로의 정확도를 대상으로 하여 결과값을 추출하였다.

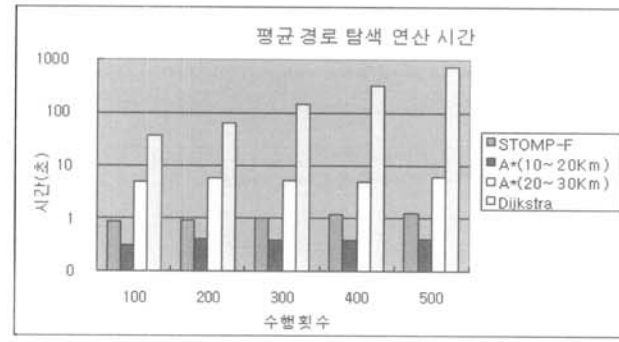
실험을 위한 시공간 이동 패턴 탐사 시스템인 STMPMiner의 개발 환경은 Windows XP기반에서 개발도구로 Eclipse

SDK 3.2.2와 JDK 1.6.0을 사용하였고, 이동 객체 데이터와 Geometry 데이터를 저장하기 위해서 Oracle 10g로 데이터베이스 시스템을 구축하였다. 최적 이동 패턴 추출 성능에 대한 실험 환경은 STMPMiner 시스템으로 Pentium Ⅳ 2.4GHz PC와 2GByte 메인 메모리를 사용하였고, 데이터베이스 시스템으로 Pentium Ⅳ 2.0GHz PC와 1GByte 메인 메모리를 사용하였다. 또한 실험을 위한 Geometry 데이터로 서울시 행정 구획 데이터와 도로 네트워크 데이터를 사용하였고, 이동 객체 데이터로 서울시의 도로 네트워크 상에서 택시들의 운행 기록을 측위하여 이력 데이터를 생성하였다. 측위된 이력 데이터는 도로 네트워크 상에서의 교차점이나 분기점, 종료점 등을 노드로 설정하여 각 노드에서 이동 객체의 위치 정보를 샘플링하였다. 다음 <표 4>는 성능 평가 실험에 이용되는 이력 데이터의 유형이고, <표 5>는 실험 데이터에 대한 특성을 나타낸다. 데이터 집합의 이름에서 N은 이동 객체 수, D는 이동기간, G는 샘플링 위치를 의미한다.

5.1 평균 경로 탐색 연산 시간의 성능 평가

경로 탐색에 있어서의 평균 연산 시간은 각 알고리즘들에 대해 공통적으로 동일한 공간 범위의 경로들을 대상으로 연산 처리 횟수 100, 200, 300, 400, 500번을 탐색하도록 하여 측정하였다. 다음 <표 6>은 STOMP-F와 Dijkstra, A* 알고리즘들에 대해서 경로 탐색에 있어서의 평균 연산 시간을 산출한 결과이다.

실험 결과 휴리스틱 가중치를 10~20Km 구간으로 지정



(그림 5) 평균 경로 탐색 연산 시간 비교

한 A* 알고리즘이 가장 좋은 연산 처리 성능을 보였으며, 다음으로 제안된 STOMP-F 알고리즘이 좋은 효율을 보였다. Dijkstra 알고리즘은 결과적으로 가장 큰 연산시간을 가지는데, 이는 Dijkstra 알고리즘이 출발지점을 중심으로한 동심원 형태로 모든 노드를 대상으로 하여 경로 탐색을 수행하기 때문이다. A* 알고리즘의 경우 구간별로 연산 시간의 차이가 크게 나타나는데, 10~20Km 구간의 연산시간이 20~30Km 구간의 연산시간보다 훨씬 좋은 성능을 보였다. 이는 A* 알고리즘의 특징인 휴리스틱 가중치에 따라 목적지 방향으로의 직진성 범위가 확대되거나 축소되어 탐색 대상 노드의 수가 많아지거나 적어져 이러한 결과가 나타난다. 즉, A* 알고리즘은 직진성이 강하게 부여되면 될수록 연산 속도와 접근 노드의 수에서 좋은 성능을 보이지만, 알고리즘의 탐색 성능은 모든 경우에 비하여 최악의 결과를 가져온다.

〈표 4〉 성능 평가를 위한 실험 데이터 유형

MOID	NODE		Valid Time
	x	y	
15440001	126.962847	37.533317	2006/10/14/11/12/02
15440001	126.970583	37.536738	2006/10/14/11/12/53
15440001	126.970708	37.541675	2006/10/14/11/13/50
15440001	127.003377	37.502183	2006/10/14/11/14/01
...	...	...	...

〈표 5〉 실험 데이터 특성

데이터 집합 이름	이동 객체 수(N)	이동기간(D)	실데이터 크기
N4000-D1014-Gnode	4000	1일	58.9 MB

〈표 6〉 평균 경로 탐색 연산 시간 실험 결과

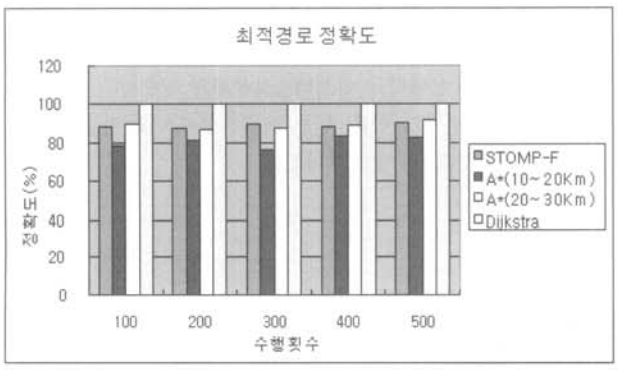
수행횟수	STOMP-F	A*(10~20Km)	A*(20~30Km)	Dijkstra
100	0.88	0.323	5.07	36.34
200	0.93	0.41	5.78	64.54
300	1.01	0.38	5.34	145.36
400	1.22	0.39	4.96	324.32
500	1.26	0.42	5.98	724.34

이에 대한 검증은 최적경로의 정확성에서 확실하게 확인할 수 있다. 즉, 휴리스틱 가중치를 어떻게 설정하느냐가 A* 알고리즘의 성능을 좌우하는 주요 요인이다. STOMP-F 알고리즘은 최대 빈발도를 탐사하는 과정을 통해 최적 경로를 추출하기 때문에 빈발 패턴을 탐사하는 시간이 길어 위와 같은 결과를 보였으며, 탐사 대상 데이터의 양에 따라 연산 시간이 좌우되는 특성이 있다.

5.2 최적 경로의 정확도에 대한 성능 평가

최적 경로의 정확도는 각 알고리즘에 의해 탐색된 경로들 얼마만큼 최적 경로에 근접했는지를 평가하는 기준으로, Dijkstra 알고리즘의 최적 경로 정확도는 100%이다. Dijkstra 알고리즘은 지도상의 모든 노드들을 대상으로 하여 경로 탐색을 수행하기 때문에 어떠한 경우에서도 최적 경로를 탐색할 수 있다. 다음 <표 7>은 STOMP-F와 Dijkstra, A* 알고리즘들에 대해서 경로 탐색에 있어서의 최적 경로 정확도를 산출한 결과이다.

실험 결과를 보면 STOMP-F 알고리즘이 A* 알고리즘에 비해 Dijkstra 알고리즘에 가장 근접하게 접근한 것을 볼 수 있다. 즉 STOMP-F 알고리즘에 의해 탐색되는 경로들은 최적 경로에 가깝다는 것을 확인할 수 있다. 이는 앞서 언급한대로 이동 객체의 경험적 이력 데이터 집합으로부터 빈발도가 높은 패턴들을 대상으로 최적 경로를 탐색하기 때문에 최종 목표에 도달하는 경로는 반드시 찾아질 수 있다. 반면 A* 알고리즘은 10~20Km 구간에서 가장 정확도가 떨어지게 되는데, 이는 휴리스틱 가중치에 의해 지정된 범위의 노드들만을 대상으로 경로 탐색을 수행하기 때문에 탐색 범위가 작아 최적 경로를 찾을 확률이 낮아진다. 따라서 휴리스틱 가중치를 넓게 지정하여 보다 정확하게 최적 경로를 탐색할 수 있지만, 앞선 실험 결과를 보면 접근 노드수가 많을 경우 연산 시간이 증가하기 때문에 적절한 휴리스틱 가중치의 설정이 중요하다.



(그림 6) 최적 경로 정확도 비교

6. 결론 및 향후 연구과제

이동 객체의 위치 이력 데이터를 이용하여 다양한 응용분야에서 활용할 수 있는 새로운 위치 기반 서비스를 개발하기 위한 목적으로 다양한 시공간 이동 패턴 탐사 기법에 대한 연구가 진행되어 왔으나, 기존의 패턴 탐사 연구를 통해 개발되어 실질적으로 실생활에 적용되고 있는 위치 기반 서비스는 아직 미비한 실정이다. 이에 본 논문에서는 방대한 이동 객체의 이력 데이터 집합으로부터 복합적인 시간 및 공간 제약을 갖는 최대 빈발 패턴을 추출하기 위한 최적 이동 패턴 탐사 문제에 대해 정의하였고, 실세계에 적용가능한 위치 기반 서비스의 응용으로 최적 이동 패턴을 추출하기 위한 STOMP-F 기법을 제시하였다. 제안된 기법은 실세계에서 특정 목적을 가진 이동 객체가 가장 효율적인 경로를 통해 이동할 수 있도록 최적 경로에 해당하는 최적 이동 패턴을 탐사하기 위한 방법으로, 특정한 지점들 사이를 이동한 객체의 이력 데이터들 중 객체가 가장 빈번하게 이동한 경로를 탐색하여 최적 경로로 결정하는 패턴 빈발도를 이용한 탐사 방법이다. 제안된 기법에서는 패턴 탐사를 수행하는데 있어 SeqExtractor 연산을 통해 이동 객체의 이력 데이터를 시퀀스로 변환시키고, 객체의 위치값과 공간영역간의 위상 관계를 고려하여 이동 객체의 위치 속성에 대한 최하위 수준에서의 공간 일반화를 통해 보다 효율적으로 패턴 탐사를 수행할 수 있도록 하였다. 또한 실제적인 빈발 이동 패턴을 탐사하는 OptPathExtractor와 Freq_link 연산을 통해 최적 이동 패턴을 추출하도록 각 알고리즘들을 설계 및 구현하였다.

STOMP-F 알고리즘에 대한 실험은 최적 경로 탐색에 일반적으로 가장 많이 사용되는 Dijkstra 알고리즘과 A* 알고리즘을 대상으로 평균적인 데이터를 산출하여 성능을 비교하였다. 최적 경로 탐색 알고리즘들의 성능을 평가하기 위한 실험 기준은 탐색 연산 시간과 최적 경로의 정확도를 대상으로 하였다. 먼저 탐색 연산 시간을 기준으로 실험한 결과, 휴리스틱 가중치를 10~20Km 구간으로 지정한 A* 알고리즘이 가장 좋은 연산 처리 성능을 보였으나 최적 경로를 탐색하지 못할 확

〈표 7〉 최적 경로 정확도 실험 결과

수행횟수	STOMP-F	A*(10~20Km)	A*(20~30Km)	Dijkstra
100	88.25	78.36	90.14	100.00
200	87.92	81.34	87.32	100.00
300	89.64	76.31	88.00	100.00
400	88.34	83.74	89.31	100.00
500	90.54	82.64	92.14	100.00

률이 높았으며, 대체로 안정적으로 최적 경로를 탐색할 확률이 높은 20~30Km 구간의 연산 처리 시간은 STOMP-F 알고리즘보다 비효율적이었다. 또한 최적 경로의 정확도에 대한 실험 결과, Dijkstra 알고리즘은 모두 정확하게 최적 경로를 탐색하며, 이를 기준으로 제안된 STOMP-F 알고리즘과 A^* 알고리즘을 비교한 결과 STOMP-F 알고리즘이 A^* 알고리즘에 비해 좋은 정확도를 보였다. 이는 이동 객체가 과거 이동했던 경험적 데이터로부터 빈발도가 높은 패턴들만을 추출하여 최적 경로를 탐색하기 때문에 최종 목표에 도달하는 경로는 반드시 찾아질 수 있기 때문이다.

향후 연구과제로는 최적 이동 경로 탐색을 위한 패턴 탐사 기법을 이용하여 단위 시간동안 이동 객체가 순회해야 하는 지점들에 대한 스케줄링 경로 예측을 위한 마이닝 기법의 개발이 필요하다.

참 고 문 헌

[1] D. O. Kim, H. K. Kang, D. S. Hong, J. K. Yun and K. J. Han, "STMPE : An Efficient Movement Pattern Extraction Algorithm for Spatio-temporal Data Mining," in proc. on International Conference on Computational Science and Its Applications(ICCSA), pp.259-269, 2006.

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

[3] H. Cao, N. Mamoulis and D. W. Cheung, "Mining Frequent Spatio-Temporal Sequential Patterns," in proc. on the 5th IEEE International Conference on Data Mining(ICDM), pp.82-89, 2005.

[4] Y. Huang, L. Zhang and P. Zhang, "Finding Sequential Patterns from a Massive Number of Spatio-Temporal Events," SDM, SIAM, 2006.

[5] J. W. Lee, O. H. Paek and K. H. Ryu, "Temporal Moving Pattern Mining for Location-Based Service," The Journal of Systems and Software, Vol.73, 2004.

[6] G. Yavas, D. Katsaros, O. Ulusoy and Y. Manolopoulos, "A Data Mining Approach for Location Prediction in Mobile Environmensts," Data & Knowledge Engineering, Vol.54, pp.121-146, 2005.

[7] S. Y. Hawng, Y. H. Liu, J. K. Chiu and E. P. Lim, "Mining Mobile Group Patterns : A Trajectory-based Approach," PAKDD, Lecture Notes in Computer Science, Springer, Vol.3518, pp.713-718, 2005.

[8] Y. Wnag, E. P. Lim and S. Y. Hwang, "On Mining Group Patterns of Mobile Users," DEXA, Lecture Notes in Computer Science, Springer, Vol.2736, pp.287-296, 2003.

[9] J. Gudmundsson, M. V. Kreveld and B. Speckmann, "Efficient Detection of Motion Patterns in Spatio-Temporal Data Sets," in proc. on the 12th annual ACM international workshop on Geographic Information Systems(GIS), pp. 250-257, 2004.

[10] P. Laube and S. Imfeld, "Analyzing Relative Motion within Groups of Trackable Moving Point Object," in GIScience, Notes in Computer Science, Springer, Vol.2478, pp.132-144, 2002.

[11] 백옥현, "위치 기반 서비스를 위한 이동 객체의 시간 패턴 탐사 기법", 충북대학교 대학원, 석사학위논문, 2002.

[12] 이준욱, "위치 기반 서비스를 위한 이동 객체의 시간 패턴 탐사", 한국정보과학회 논문지, 제29권, 제5호, 2002.

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

[14] 한선영, "시공간 이동 시퀀스 패턴 마이닝 기법", 이화여자대학교 대학원, 석사학위논문, 2006.

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

[16] 고현, 김광종, 이연식, "R^*-Tree와 Grid를 이용한 이동 객체의 위치 일반화 기법", 한국컴퓨터정보학회 논문지, Vol. 12, No.2, pp.231-242, 2007.

이 연 식

e-mail : yslee@kunsan.ac.kr

1982년 전남대학교 전자계산학과(학사)

1984년 전남대학교 전자계산학과(이학석사)

1994년 전북대학교 전산응용공학과(공학박사)

1995년~1997년 군산대학교 교무부처장

1997년~1998년 University of Missouri 교환교수

1999년~2001년 군산대학교 전자계산소 소장

2004년~2005년 Ohio State University 교환교수

1986년~현 재 군산대학교 컴퓨터정보공학과 교수

관심분야 : 언어 번역기 이론, 능동 객체지향시스템, 지능형 에이전트, 마이닝 에이전트

고 현

e-mail : khyun001@kunsan.ac.kr

2001년 군산대학교 컴퓨터정보과학과(학사)

2003년 군산대학교 컴퓨터정보과학과 (이학석사)

2007년 군산대학교 컴퓨터정보과학과 (이학박사)

2008년~현 재 한국항공우주연구원 프로젝트연구원

관심분야 : 에이전트, 이동객체시스템, 시공간데이터베이스, 데이터마이닝