KIPS Trans. Softw. and Data Eng.
Vol.8, No.12 pp.473~482 pISSN: 2287-5905

https://doi.org/10.3745/KTSDE.2019.8.12.473

A Technique for Detecting Companion Groups from Trajectory Data Streams

Suhyun Kang† · Ki Yong Lee††

ABSTRACT

There have already been studies analyzing the trajectories of objects from data streams of moving objects. Among those studies, there are also studies to discover groups of objects that move together, called companion groups. Most studies to discover companion groups use existing clustering techniques to find groups of objects close to each other. However, these clustering-based methods are often difficult to find the right companion groups because the number of clusters is unpredictable in advance or the shape or size of clusters is hard to control. In this study, we propose a new method that discovers companion groups based on the distance specified by the user. The proposed method does not apply the existing clustering techniques but periodically determines the groups of objects close to each other, by using a technique that efficiently finds the groups of objects that exist within the user-specified distance. Furthermore, unlike the existing methods that return only companion groups and their trajectories, the proposed method also returns their appearance and disappearance time. Through various experiments, we show that the proposed method can detect companion groups correctly and very efficiently.

Keywords : Trajectory Data Stream, Companion Groups Detection, Stream Data Mining

궤적 데이터 스트림에서 동반 그룹 탐색 기법

강 수 현† · 이 기 용††

요 약

이동 객체의 데이터 스트림으로부터 객체들의 궤적을 분석하는 연구는 이미 이루어진 바가 있다. 그 중 같이 움직이는 객체들의 그룹, 즉 동반 그룹을 찾는 연구도 이미 존재한다. 이들 대부분은 서로 가까이 존재하는 객체들의 그룹을 탐색하기 위해 기존의 클러스터링 기법을 사용한다. 하지만 클러스터링에 기반한 방법들은 정확한 클러스터의 수를 미리 알 수 없거나 클러스터의 모양이나 크기를 제어할 수 없기 때문에 정확한 동반 그룹을 찾기 어려운 경우가 많다. 본 논문은 실시간으로 유입되는 궤적 데이터 스트림에서 기존의 클러스터링 기법이 아니라 사용자가 지정한 거리를 기반으로 동반 그룹을 탐색하는 새로운 방법을 제안한다. 본 논문에서 제안하는 기법은 서로 가까이 존재하는 객체들의 그룹을 주기적으로 탐색하며, 이 때 사용자가 지정한 거리 내에 존재하는 객체들의 그룹을 매우 효율적으로 찾아내는 기법을 사용한다. 또한 동반 그룹 및 그의 궤적만을 반환하는 기존 방법과 달리 제안 방법은 동반 그룹의 생성 시간과 지속 시간도 같이 알려준다. 본 논문에서는 다양한 실험을 통해 제안 방법이 동반 그룹을 정확하고 매우 효율적으로 탐색할 수 있음을 보인다.

키워드 : 궤적 데이터 스트림, 동반 그룹 탐색, 데이터 스트림 마이닝

1. 서 론

최근 들어 실시간으로 흘러 들어오는 데이터에 대한 분석들이 많이 이루어지고 있다. 이렇게 실시간으로 흘러 들어오는 데이터들 중 움직이는 객체들의 위치 및 시간 정보를 가지고 있는 데이터를 궤적 데이터 스트림이라고 한다. Fig. 1은 허리케인의 위치 및 시간 정보를 나타내는 궤적 데이터 스트림으로부터 허리케인의 궤적을 추출한 예이다. 지금까지 이러한 궤적 데이터 스트림을 분석하기 위한 다양한 연구들이 이루어져 왔다. 그의 예로서 객체들의 대표적인 이동 패턴을 탐색하거나 빈번히 발생하는 부분 궤적들을 찾는 연구들이 많이 진행되었다[1, 2]. 이러한 연구들은 교통 흐름을 분석하거나 사람들의 여행 경로 등을 분석하는데 사용될 수 있다.

지금까지 각 객체의 위치 및 시간 정보를 가지고 있는 데이터로부터 객체들의 이동 패턴 혹은 궤적을 찾는 연구는 이미 많이 이루어졌다. 이 중 많은 연구들은 각 객체의 위치 및

※ 이 논문은 2019년도 정부(교육부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임(No.2018R1D1A1B07045643).
† 준 회 원 : 숙명여자대학교 컴퓨터과학과 석사
†† 종신회원 : 숙명여자대학교 소프트웨어학부 교수
Manuscript Received : July 25, 2019
First Revision: September 24, 2019
Accepted : October 15, 2019
* Corresponding Author : Ki Yong Lee(kiyonglee@sookmyung.ac.kr)

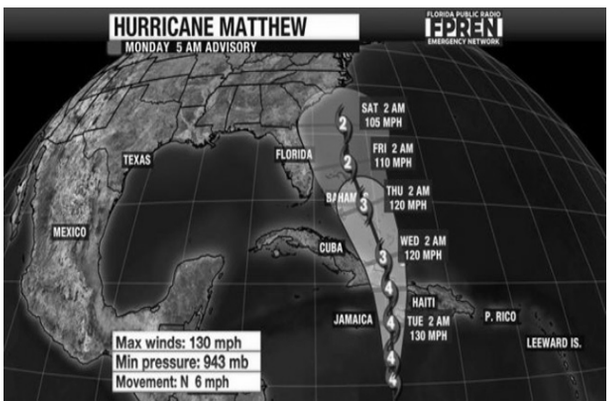

Fig. 1. Examples of Trajectory Extracted from the Trajectory Data Stream[1)]

시간이 실시간으로 흘러 들어오는 데이터 스트림이 아닌, 모든 데이터가 한꺼번에 주어진 정적(static) 데이터를 대상으로 하고 있다[5, 6]. 이에 비해 각 객체의 위치 및 시간이 실시간으로 흘러 들어오는 궤적 데이터 스트림으로부터 객체들의 궤적을 분석하는 연구는 아직 활발히 연구가 진행되는 중이다. 궤적 데이터 스트림으로부터 객체들의 궤적을 분석하는 연구에는 크게 두 가지 종류가 있다. 첫 번째는 많은 객체들의 궤적들로부터 대표적인 궤적 혹은 부분 궤적들을 탐색하는 연구이다[10, 11]. 이들은 교통량이 많이 발생하는 경로 등을 파악하는 데 사용될 수 있다. 두 번째는 같이 이동하는 객체들을 탐색하는 연구이다[14-16]. 같이 이동하는 객체들을 동반 그룹(companion groups)이라 하며, 이들은 같이 여행하는 관광객 등을 탐색하는 데 사용될 수 있다. 본 논문에서는 이 중 두 번째 종류인 동반 그룹을 탐색하는 문제를 다룬다.

궤적 데이터 스트림에서 동반 그룹 탐색의 응용 분야는 다양하게 존재한다. 카풀을 위해 같은 이동 경로를 가진 사람을 탐색하는 것이 그 예이다. 비슷한 예로 동물들의 이동 경로 또한 많은 연구 분야에서 사용될 수 있다. 이렇게 다양한 분야에서 궤적 데이터 스트림을 통한 동반 그룹 탐색이 응용될 수 있다.

궤적 데이터 스트림으로부터 동반 그룹을 탐색하는 방법들은 서로 가까이 존재하는 객체들을 주기적으로 탐색한다. 이를 위해 대부분의 연구들은 k-means나 DBSCAN 등과 같은 기존의 클러스터링(clustering) 기법을 사용한다[16-18]. 하지만 클러스터링 기법을 사용하여 같이 움직이는 그룹을 탐색하는 것은 다음과 같은 문제점이 있다. 첫째, 많은 클러스터링 알고리즘은 k-means와 같이 클러스터의 수를 미리 지정해야 하거나, DBSCAN과 같이 클러스터를 구성하는 객체들의 최소 밀도를 미리 지정해야 한다. 하지만 이러한 값들을 미리 정확히 아는 것은 매우 어렵다. 둘째, 대부분의 클러스터링 알고리즘은 그의 결과로 생성될 클러스터의 모양이나 크기를 제어하기 어렵다. 따라서 매우 먼 거리에 존재하는 객체들도 하나의 클러스터에 속해 동반 그룹으로 잘못 판정될 수 있다. 따라서 본 논문은 정확히 사용자가 지정한 거리 내에 존재하는 객체들만을 동반 그룹으로 탐지하는 새로운 방법을 제안한다.

제안 방법은 사용자로부터 매개변수로 3개의 값 T_{size}, $T_{distance}$, $T_{duration}$을 받는다. 제안 방법은 어떤 그룹의 객체 수가 T_{size}개 이상이고, 모든 객체 간의 거리가 서로 $T_{distance}$ 이하이며, 그룹의 유지 시간이 최소 $T_{duration}$일 때 해당 그룹을 동반 그룹으로 탐지한다. 따라서 제안 방법은 사용자가 원하는 동반 그룹을 정확히 탐색할 수 있다. 또한 제안 방법은 탐지된 동반 그룹과 그의 궤적만을 반환하는 기존 방법들과 달리, 동반 그룹이 생성된 시간과 지속 시간까지 반환해준다. 이를 위해 제안 방법은 서로 거리가 $T_{distance}$ 내에 존재하는 객체들의 그룹들을 주기적으로 탐색하는 한편, 해당 그룹들이 유지되는 시간에 대한 정보를 별도로 관리한다.

본 논문은 궤적 데이터 스트림으로부터 동반 그룹을 탐색하는 새로운 방법을 제안한다. 본 논문의 기여는 다음과 같다: (1) 본 논문은 서로 거리가 $T_{distance}$이하인 객체들의 그룹을 빠르게 탐색할 수 있는 새로운 방법을 제안한다. (2) 본 논문은 기존 방법과 달리 동반 그룹의 생성 시간과 지속 시간까지 제공하는 새로운 방법을 제안한다. (3) 본 논문은 다양한 실험을 통해 제안 방법의 정확성 및 효율성을 보인다.

본 논문의 구성은 다음과 같다. 2장에서는 관련 연구들을 살펴보고, 3장에서는 문제 정의를 한 뒤, 4장에서는 제안 방법을 상세히 설명한다. 5장에서는 제안 방법의 실험 결과를 보이며, 6장에서는 결론을 맺는다.

2. 관련 연구

주어진 데이터로부터 객체들의 이동 패턴 혹은 궤적을 분석하는 연구는 크게 정적 데이터에 대한 연구와 데이터 스트림에 대한 연구로 나눌 수 있다.

정적 데이터에서 객체들의 궤적을 분석하는 연구는 굉장히 많이 이루어져 왔다. [1]은 유사한 궤적들의 그룹을 자동으로 탐색해 준다. [2]는 두 개의 궤적 사이에 특정 시간 내에서만 유사한 부분을 탐색한다. [3, 4]는 궤적 전체를 보고 전체 궤적이 유사한 궤적들을 모아 클러스터링 해준다. 하지만 궤적들은 각 객체 마다 매우 다양한 모습을 가지기 때문에 전체가 유사한 궤적을 찾는 다는 것은 한계가 있다. 그래서 궤적 전체가 아닌 부분적으로 유사한 궤적을 탐색하는 연구들도 이루어져 왔다. [5]는 각 궤적을 작은 선분들로 끊어주고 유사한 선분들을 그룹화여 유사한 부분 궤적을 탐색한다. 이 외에도 같이 이동하는 객체들을 탐색하는 연구들도 진행되었다. [6]은 특정 반경을 가지는 원 내에 일정 시간 이상동안 같이 존재하는 객체들의 무리를 탐색한다. 하지만 제안 방법은 일정 크기의 원 내에 존재하는 객체들이 아니라 모든 객체 간의

1) https://news.wjct.org/post/gov-scott-east-coast-residents-leave-now-hurricane-matthew-hits

거리가 $T_{distance}$ 이하인 그룹을 탐색한다는 차이가 있다. 특정 크기를 가진 원 내의 존재 여부로 무리의 지속 여부를 탐색하게 되면 함께 움직이는 객체가 조금만 원을 벗어나게 되도 이 객체를 놓칠 수 있다는 문제가 있다. 그래서 [7, 8]은 이를 극복하기 위해 특정 반경을 가진 원이 아니라 임의의 모양으로 밀집되어 무리를 이루는 그룹의 지속 여부를 탐색한다.

데이터 스트림에서 객체들의 궤적을 분석하는 연구는 유사한 궤적들을 탐색하여 대표적인 궤적들을 출력해주는 연구와 함께 움직이는 객체, 즉 동반 그룹을 탐색하는 연구 크게 두 종류로 구분해 볼 수 있다. 1장에서 언급한 바와 같이 궤적 데이터 스트림에서 유사한 궤적을 그룹화하여 대표적인 궤적을 출력해주는 연구들이 이미 진행되었다. [9]는 각 시간 간격에서 연속적인 궤적 선분(line segment)들을 클러스터링하여 궤적의 대표 궤적을 유지하고 유지된 궤적을 추출한다. [10]은 움직이는 객체들의 궤적 선분들을 서로 가까운 궤적 선분들끼리 그룹화하여 소그룹을 형성한다. 그리고 새로운 궤적 선분이 들어오면 그 선분을 가장 유사한 소그룹으로 삽입하여 새로운 선분이 추가된 전체 궤적을 반환해 준다. [11] 또한 [10]와 유사하게 객체들의 부분 궤적(sub-trajectory)을 서로 가까운 부분 궤적들끼리 소그룹으로 그룹 화한다. 하지만 [11]은 소그룹을 가장 잘 나타낼 수 있는 대표 궤적을 기반으로 각 시간 간격에서 부분 궤적과 대표 궤적간의 유사도를 파악하여 가장 유사한 그룹으로 삽입하여 그룹을 유지한다. 그리고 작은 그룹들을 가지고 다시 한 번 클러스터링을 수행하여 모든 궤적을 반환해준다. [12]는 개별적인 궤적 인덱스를 사용하여 유사한 궤적끼리 인덱스를 사용하여 정렬을 시켜 주고 저장된 모든 인덱스의 전체 목록을 반환해준다. [13]은 부분 궤적을 공간, 시간, 방향, 의미있는 위치 정보를 기반으로 유사한 움직임을 가지는 부분 궤적을 클러스터링하여 궤적을 탐색한다.

이 외에도 궤적 데이터 스트림에서 같이 움직이는 객체들의 그룹을 출력해주는 연구들이 있다[14]. 이렇게 같이 움직이는 객체들의 그룹을 동반 그룹이라 부른다. [15]은 궤적을 시간(t) 및 위치(x, y) 좌표를 사용하여 최소 경계 상자(minimal bounding box, MBB)로 나타내고, (x, y, t) 좌표가 모두 유사한 MBB들을 탐색하여 MBB가 유사한 객체들을 반환해 준다. [16]는 서로 근접한 객체들의 그룹을 주기적으로 탐색하여 일정 시간 이상 그룹을 형성하는 객체들을 동반 그룹으로 반환한다. 이 방법은 주어진 시간에 서로 근접한 객체들의 그룹을 탐색하기 위해 밀도 기반 클러스터링 기법을 사용한다. 이 때 클러스터링 비용을 줄이기 위해 원 객체들을 대상으로 클러스터링 하는 것이 아니라 가까운 객체들을 버디(buddy)라는 작은 그룹들로 먼저 그룹화하고 버디 단위로 클러스터링을 실행한다. [17, 18] 또한 버디 대신 소그룹(micro-group)이라는 용어를 정의하여 사용하였지만 [14]와 유사하게 주기적으로 소그룹 단위로 클러스터링을 수행하고 일정 시간 유지된 그룹을 반환해준다.

앞서 설명한 바와 같이 데이터 스트림에서 동반 그룹을 탐색하는 대부분의 연구들은 일정 시점에 서로 가까이 존재하는 객체들의 그룹을 탐색하기 위해 기존의 클러스터링 기법을 사용한다[16-18]. 하지만 클러스터링 기법을 사용하여 동반 그룹을 탐색하는 경우 적절한 클러스터의 수 또는 클러스터의 최소 밀도와 같이 클러스터링에 필요한 매개변수의 값을 미리 알기 어렵다는 문제가 있다. 또한 클러스터링의 결과로 생성되는 클러스터의 모양 또는 크기를 제어하기 어렵기 때문에, 비교적 먼 거리에 존재하는 객체들도 하나의 큰 클러스터에 속하는 경우 동반 그룹으로 잘못 판정될 수 있다. 따라서 본 논문은 궤적 데이터 스트림에서 동반 그룹을 탐색할 때, 기존의 클러스터링 기법 대신 사용자가 지정한 거리 이하로 서로 가까이 존재하는 객체들의 그룹을 탐색하는 새로운 방법을 제안한다.

본 논문에서 제안하는 방법은 사용자가 지정한 거리 이하로 서로 가까이 존재하는 객체들의 그룹을 효율적으로 탐색하기 위해 기존의 연관 규칙 탐색 알고리즘인 Apriori 알고리즘[19, 20]과 유사한 방법을 사용한다. 또한 본 논문에서 제안하는 방법은 기존 방법과 달리 동반 그룹의 생성 시간과 지속 시간도 추가적으로 출력해준다.

3. 문제 정의

본 논문에서 가정하는 궤적 데이터 스트림은 Fig. 2와 같은 형태를 가진다. 여기서 각 레코드는 각 객체의 위치 및 그 때의 시간 정보를 나타낸다. $object_{ID_i}$는 해당 객체의 ID, $time_i$는 시간, x_i, y_i는 해당 객체의 위치를 나타낸다.

동반 그룹을 탐색하는 세부적인 방법을 설명하기에 앞서 본 논문에서 사용될 중요한 개념들을 Fig. 3을 통해 간단히 설명한다.

$$
\begin{gathered}
(object\_ID_1, time_1, x_1, y_1)\\
(object\_ID_2, time_2, x_2, y_2)\\
(object\_ID_3, time_3, x_3, y_3)\\
(object\_ID_4, time_4, x_4, y_4)\\
(object\_ID_5, time_5, x_5, y_5)\\
(object\_ID_6, time_6, x_6, y_6)\\
\vdots\\
(object\_ID_i, time_i, x_i, y_i)\\
\vdots
\end{gathered}
$$

Fig. 2. Trajectory Data Stream

- 그룹: 객체 간 최대 거리 $T_{distance}$가 주어졌을 때, 그룹은 어떤 시점을 기준으로 서로 $T_{distance}$ 이하 내에 존재하는 객체들로만 이루어진 집합을 의미한다. 예를 들어, Fig. 3에서 첫 번째 시간 간격 $[t_0, t_i]$ 사이에서 객체 o_1, o_2, o_3, o_4, o_5들은 서로 거리가 $T_{distance}$ 이하므로 그룹 g_1을 형성하고, 객체 o_7, o_8, o_9, o_{10}들도 서로 거리가 $T_{distance}$ 이하이므로 그룹 g_2를 형성한다.

- 그룹 크기: 그룹의 크기는 해당 그룹에 속한 객체들의 수

를 의미한다. 예를 들어, Fig. 3에서 g_1의 크기는 5이고 g_2의 크기는 4이다.

- 그룹 생성: 그룹 생성이란 특정 객체들이 이전 시점에 없었던 그룹을 새로 형성하는 것을 의미한다. 예를 들어, Fig. 3에서 t_0이 최초 시간이라 할 때, g_1과 g_2는 시간 간격 $[t_0, t_i]$ 사이에서 새로 형성된 그룹으로 볼 수 있다.

- 그룹 지속 시간: 그룹의 지속 시간은 해당 그룹이 생성된 후부터 계속 유지되고 있는 시간을 의미한다. 예를 들어, Fig. 3에서 g_1이 t_0에서 생성되었다고 하고 현재 시점이 t_4라고 하자. g_1을 구성하는 객체들은 t_4까지도 계속 그룹을 유지하고 있으므로 t_4를 기준으로 했을 때 g_1의 지속 시간은 $t_4 - t_0$이다.

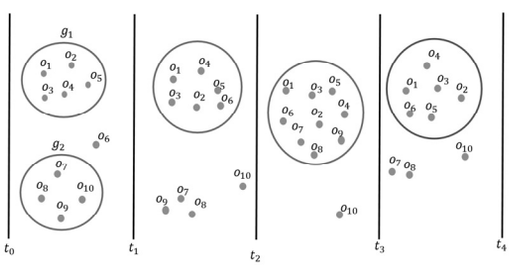

Fig. 3. Examples of Companion Groups

정의 1 (그룹) : 객체 간의 최대 거리 $T_{distance}$가 주어졌을 때, 만약 어떤 시점 t에 공존하는 객체들의 집합 $O=\{o_1,o_2,...,o_n\}$가 다음 조건을 만족하면 O를 그룹이라 한다.

$$\forall o_i, o_j \in O,\ dist(o_i, o_j) \le T_{distance}$$

단, t에서 o_i의 위치를 (x_i,y_i)라하고 o_j의 위치를 (x_j,y_j)라 할 때, $dist(o_i,o_j)=\sqrt{(x_j-x_i)^2+(y_j-y_i)^2}$로 정의한다.

정의 2 (동반 그룹) : 그룹 최소 크기 T_{size}와 그룹 최소 지속 시간 $T_{duration}$이 주어졌을 때, 어떤 그룹 g가 다음 조건을 만족하면 g를 동반 그룹이라 한다.

$$size(g) \ge T_{size}$$
$$duration(g) \ge T_{duration}$$

단, $size(g)$는 그룹 g의 크기를 나타내며, $duration(g)$은 현재 시점을 기준으로 g의 지속시간을 나타낸다.

본 논문에서 고려하는 문제는 다음과 같이 나타낼 수 있다. 각 레코드가 $(o_i,\ t_i,\ x_i,\ y_i)$ 형태인 궤적 데이터 스트림을 D라고 하자. 여기서 o_i는 객체의 ID, t_i는 레코드가 생성된 시간, x_i와 y_i는 t_i 시점에 객체 o_i의 위치를 나타낸다. 본 논문에서 고려하는 문제는 데이터 스트림 D로부터 정의 2에서 정의한 동반 그룹을 탐색하는 문제이다.

4. 제안 방법

본 논문에서 제안하는 방법은 서로 가까이 존재하는 객체들의 그룹을 주기적으로 탐색한다. 이 때 제안 방법은 클러스터링 기법을 사용하는 기존 방법과 달리 서로 거리가 $T_{distance}$ 이내에 존재하는 객체들의 그룹을 탐색한다. 여기서 $T_{distance}$는 사용자에게 받는 임의의 값이다. 만약 사용자가 확실하고 밀접한 동반 그룹을 탐색하고 싶다면 $T_{distance}$를 작게 설정하고 광범위한 동반 그룹을 탐색하고자 한다면 $T_{distance}$를 크게 설정한다. 이때 $T_{distance}$의 구체적인 값은 시스템에서 자동으로 제시하기 어려운 값이며, 사용자의 응용에 따라 판단해야 한다. 이렇게 탐색된 그룹들의 정보를 유지하다가 그의 크기가 T_{size} 이상이고, 지속 시간이 $T_{duration}$ 이상인 경우 해당 그룹을 동반 그룹으로 출력한다. 이 때 제안 방법은 해당 동반 그룹이 생성된 시간과 지속 시간을 같이 출력해 준다. 본 장에서는 제안 방법을 자세히 설명한다.

제안 방법은 $\triangle t$라는 시간 간격마다 가까이 존재하는 객체들의 그룹을 주기적으로 탐색한다. 본 논문에서는 $t_i = t_{i-1} + \triangle t$이며 매 t_i마다 그룹 탐색이 수행된다고 가정한다. 각 시점 t_i에는 $[t_{i-1},\ t_i]$ 사이에 유입된 레코드들에 포함된 객체들에 대해 그룹 탐색을 진행한다. 만약 $[t_{i-1},\ t_i]$사이에 동일한 객체에 대해 여러 레코드들이 존재하는 경우, 이들의 평균 좌표를 계산하여 해당 객체의 좌표로 가정한다.

4.1 단순 방법

$[t_{i-1},\ t_i]$ 사이에 유입된 레코드들에 포함된 모든 객체들을 $O=\{o_1,\ o_2,\ ...,\ o_n\}$이라 하자. t_i에는 이들 중 서로 거리가 $T_{distance}$ 이내에 존재하는 객체들의 그룹을 탐색한다. 이러한 그룹을 찾는 가장 간단한 방법 중 하나는 다음과 같다. 먼저 O에서 $dist(o_i,\ o_j) \le T_{distance}$를 만족하는 객체들의 쌍 $o_i,\ o_j$를 모두 찾아 (단, $o_i \ne o_j$) 이들로부터 두 개의 객체로 이루어진 그룹 $\{o_i,\ o_j\}$을 생성한다. 두 개의 객체로 이루어진 그룹들이 모두 생성되고 나면, 이제 이 그룹들의 크기를 하나 씩 확장해 나간다. 만약 $dist(o_i,\ o_j) > T_{distance}$이면 o_i와 o_j 모두를 포함하는 집합은 절대로 그룹을 형성할 수 없기 때문에 $\{o_i,\ o_j\}$은 앞으로의 계산에서 제외된다. 앞에서 구해진 두 개의 객체로 이루어진 각 그룹 $\{o_i,\ o_j\}$에 대해 O에서 o_i와 o_j를 제외한 각 객체 o_k에 대해 $dist(o_i,\ o_k) \le T_{distance}$, $dist(o_j,\ o_k) \le T_{distance}$를 만족하면 이들로부터 세 개의 객체로 이루어진 그룹 $\{o_i,\ o_j,\ o_k\}$를 생성한다. 세 개의 객체로 이루어진 그룹들이 모두 생성되었다고 하자. 이제 각 그룹 $\{o_i,\ o_j,\ o_k\}$에 대해 O에서 $o_i,\ o_j,\ o_k$를 제외한 각 객체 o_l에 대해 $dist(o_i,\ o_l) \le T_{distance}$, $dist(o_j,\ o_l) \le T_{distance}$, $dist(o_k,\ o_l) \le T_{distance}$를 만족하면 이들로부터 네 개의 객체로

이루어진 그룹 $\{o_i, o_j, o_k, o_l\}$을 생성한다. 이렇게 더 이상 그룹이 확장되지 않을 때까지 확장을 진행한다. 확장이 종료되고 나면 생성된 그룹들 중 크기가 T_{size} 이상이고 다른 그룹에 포함되지 않는 그룹들만을 탐색된 그룹으로 반환한다.

하지만 이렇게 기존 그룹에 객체를 하나씩 추가하여 확장해 나가는 방식은 매 단계마다 각 그룹에 대해 해당 그룹에 속하지 않은 모든 객체들과의 조합을 고려해야 한다. 따라서 객체 수가 증가할수록 매우 비효율적이다. 또한 어떤 그룹에 객체를 하나 추가하기 위해서는 해당 그룹 내의 모든 객체와 새 객체 간의 거리를 모두 비교해야 한다. 따라서 본 논문에서는 이렇게 각 그룹에 객체를 하나씩 추가하여 확장해 나가는 대신 더 효율적으로 그룹을 확장하여 탐색하는 방법을 제안한다.

4.2 동반 그룹 탐색

본 논문에서 제안하는 동반 그룹 탐색 방법은 크게 (1) 주기적 그룹 탐색과 (2) 지속성 판단을 통한 동반 그룹 탐색 2단계로 구성된다. 첫 번째 단계는 서로 거리가 $T_{distance}$ 이내에 존재하는 객체들의 그룹들을 주기적으로 탐색하는 단계이다. 두 번째 단계는 첫 번째 단계에서 탐색된 그룹들에 대해 크기 및 지속 시간을 체크하여 최종적으로 동반 그룹을 반환하는 단계이다. Fig. 3은 이 두 단계를 통해 동반 그룹 $\{o_1, o_2, o_3, o_4, o_5\}$을 찾은 예를 나타낸다. 아래에서는 이 두 단계를 순서대로 설명한다.

1) 주기적 그룹 탐색

현재 시점을 t_i라고 할 때, 이 단계에서는 $[t_{i-1}, t_i]$ 사이에 유입된 모든 객체들 $O=\{o_1, o_2, ..., o_n\}$에 대한 그룹을 탐색한다. 이 때 4.1절에서 설명한 단순 방법을 사용하면 매우 비효율적이므로 제안 방법은 보다 효율적인 방법을 사용한다. 제안 방법은 각 그룹에 객체를 하나씩 추가하여 그룹을 확장해 나가는 대신, 크기가 k인 그룹 2개를 병합하여 크기가 $k+1$인 그룹을 생성한다. 이는 연관 규칙 탐색에 널리 사용되는 Apriori 알고리즘과 기본 전략이 유사하다. 아래에서 제안 방법을 좀 더 자세히 설명한다.

먼저 제안 방법은 단순 방법과 동일하게 O에서 $dist(o_i, o_j) \le T_{distance}$를 만족하는 객체들의 쌍 o_i, o_j를 모두 찾아 (단, $o_i \neq o_j$) 이들로부터 두 개의 객체로 이루어진 그룹 $\{o_i, o_j\}$를 생성한다. 두 개의 객체로 이루어진 그룹들이 모두 생성되고 나면, 제안 방법은 다음과 같이 추가적인 그룹 제거 작업을 수행한다. 두 개의 객체로 이루어진 그룹들에 포함되어 있는 각 객체 o_i에 대해 그들이 속한 그룹의 총 개수를 센다. 만약 이 개수가 $T_{size}-1$ 미만이라면 o_i를 포함하는 그룹의 크기는 절대 T_{size} 이상이 될 수 없음을 뜻한다. 따라서 o_i를 포함한 그룹들을 앞으로의 그룹 탐색에서 제외한다. 예를 들어, Fig. 4와 같이 $O=\{o_1, o_2, o_3, o_4, o_5\}$로부터 두 개의 객체로 이루어진 그룹 5개가 생성되었다고 하자. 여기서 o_1, o_2, o_3, o_4, o_5는 각각 2, 3, 3, 1, 1개의 그룹에 속해있다. 따라서 만약 $T_{size} = 3$이 라면, o_4와 o_5를 포함하고 있는 그룹의 개수는 각각 1개이다. 그러므로 이는 2보다 작기 때문에 절대 그룹의 크기가 3 이상이 될 수 없으므로 이들을 포함하고 있는 그룹 $\{o_2, o_5\}$, $\{o_3, o_4\}$를 그룹에서 제거한다. 이러한 과정을 통하여 불필요한 계산을 줄이고 동반 그룹이 될 가능성이 높은 그룹만을 더 효율적으로 탐색할 수 있다.

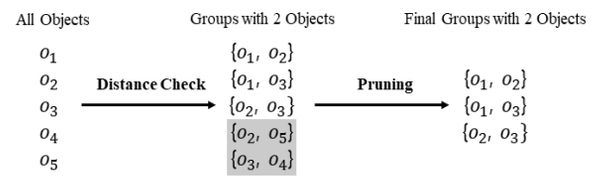

Fig. 4. Examples of Group Removal Operation

이렇게 두 개의 객체로 이루어진 그룹들을 생성한 뒤 불필요한 그룹들을 제거하고 나면, 이제 세 개의 객체로 이루어진 그룹으로 그룹을 확장해 나간다. 이 때 제안 방법은 단순 방법과 같이 두 개의 객체로 이루어진 그룹에 하나의 객체를 추가하여 세 개의 객체로 이루어진 그룹을 생성하는 것이 아니라, 두 개의 객체로 이루어진 그룹들을 병합하여 세 개의 객체로 이루어진 그룹을 생성한다. 두 그룹 $\{o_i, o_j\}$와 $\{o_i, o_k\}$를 고려하자. 이렇게 두 그룹이 o_i를 동일하게 포함하고 있으며 o_j와 o_k만 다르게 포함하고 있다면, 제안 방법은 $dist(o_j, o_k) \le T_{distance}$가 만족되는지를 체크한다. 만약 이 조건이 만족되면 제안 방법은 두 그룹을 병합하여 세 개의 객체로 이루어진 그룹 $\{o_i, o_j, o_k\}$를 생성한다. Fig. 5는 그룹을 병합하는 예를 나타낸다. 먼저 Fig. 5의 ①의 왼쪽과 같이 두 개의 객체로 이루어진 그룹 $\{o_1, o_2\}$, $\{o_1, o_3\}$, $\{o_1, o_5\}$이 존재할 때, $dist(o_2, o_3) \le T_{distance}$이 만족되어 제안 방법은 Fig. 5의 ①의 오른쪽과 같이 $\{o_1, o_2\}$와 $\{o_i, o_3\}$을 병합하여 $\{o_1, o_2, o_3\}$, $dist(o_3, o_5) \le T_{distance}$를 만족하여 $\{o_1, o_3, o_5\}$ 생성한다. 하지만 $dist(o_2, o_5) > T_{distance}$이기 때문에 $\{o_1, o_2\}$과 $\{o_1, o_5\}$는 병합되지 않는다. 이러한 방법은 세 개의 객체로 이루어진 그룹들을 병합하여 네 개의 객체로 이루어진 그룹으로 확장할 때도 똑같이 적용된다. 즉, 두 그룹 $\{o_i, o_j, o_k\}$와 $\{o_i, o_j, o_l\}$을 고려하자. 이렇게 두 그룹이 o_i, o_j를 동일하게 포함하고 있으며 o_k와 o_l 하나만 다르게 포함하고 있다면, 제안 방법은 $dist(o_k, o_l) \le T_{distance}$이 만족되는지를 체크한다. 만약 이 조건이 만족되면 제안 방법은 두 그룹을 병합하여 네 개의 객체로 이루어진 그룹 $\{o_i, o_j, o_k, o_l\}$을 생성한다. Fig. 5의 ②의 왼쪽과 같이 세 개의 객체로 이루어진 그룹 $\{o_1, o_2, o_3\}, \{o_1, o_2, o_4\}, \{o_1, o_2, o_5\}$이 존재할 때, $dist(o_4, o_5) \le T_{distance}$가 만족하고, $dist(o_3, o_5) > T_{distance}$, $dist(o_3, o_4) > T_{distance}$가 되면 제안 방법은 Fig. 5의 ②의 오른쪽과 같이 $\{o_1, o_2, o_4\}$과 $\{o_1, o_2, o_5\}$만을 병합하여 $\{o_1, o_2, o_4, o_5\}$를 생성한다. 여기서 $dist(o_1, o_5) \le T_{distance}$과 $dist(o_2, o_5) \le T_{distance}$ 등과 같은 조건은 이미 이전에 만족됨이

확인되었기 때문에 다시 확인할 필요가 없음에 주목하라. 따라서 제안 방법은 조건 하나만 확인함으로써 매우 효율적으로 두 그룹을 병합할 수 있다. 제안 방법은 이러한 방식을 사용하여 그룹이 더 이상 확장되지 않을 때까지 확장을 진행한다. 일반적으로 k개의 객체로 이루어진 그룹 $\{o_1, o_2, \cdots, o_{k-1}, o_k\}$과 $\{o_1, o_2, \cdots, o_{k-1}, o_l\}$을 고려하자. 이렇게 두 그룹이 $k-1$개의 객체를 동일하게 포함하고 있으며 o_k와 o_l 하나만 다르게 포함하고 있다면, 제안 방법은 $dist(o_k, o_l) \leq T_{distance}$이 만족되는지를 체크한다. 만약 이 조건이 만족되면 제안 방법은 두 그룹을 병합하여 $k+1$개의 객체로 이루어진 그룹 $\{o_1, o_2, \cdots, o_{k-1}, o_k, o_l\}$을 생성한다. 이렇게 확장이 종료되고 나면 생성된 그룹들 중 크기가 T_{size} 이상이고 다른 그룹에 포함되지 않는 그룹들만을 탐색된 그룹으로 반환한다.

$k+1$개의 객체로 이루어진 그룹을 생성할 때, 단순 방법은 k개의 객체로 이루어진 그룹들과 1개의 객체들 간의 모든 조합을 고려해야 한다. 반면에 제안 방법은 k개의 객체로 이루어진 그룹들 중 $k-1$개의 객체를 공통적으로 가지고 있는 그룹들 간에만 병합을 고려한다. 따라서 고려해야 할 후보의 수를 대폭 감소시킴으로써 그룹을 매우 효율적으로 탐색할 수 있다.

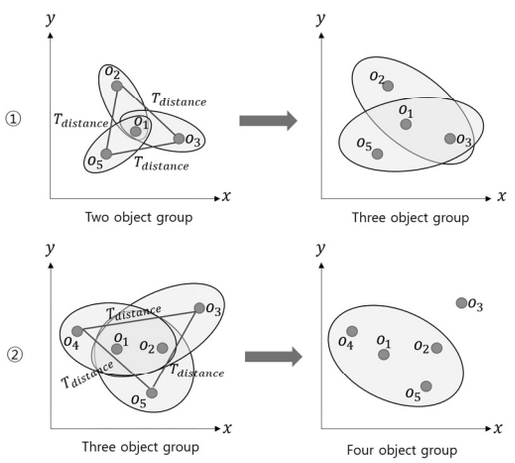

Fig. 5. Examples of Object Group Expansion

Fig. 6은 주기적 그룹 탐색의 의사코드이다. Fig. 6에서 2 - 10번째 줄은 두 개의 객체에 대해 거리를 계산하여 두 개의 객체로 이루어진 그룹을 생성한 뒤, 이들 중 불필요한 그룹을 제거하는 부분이다. 14 - 20번째 줄은 k개의 객체로 이루어진 그룹들을 병합하여 $k+1$개의 객체로 이루어진 그룹들을 생성하는 단계를 나타낸다. 21 - 26번째 줄은 탐색된 그룹 들 중 크기가 T_{size} 이상이고 다른 그룹에 포함되지 않는 그룹들만을 추출하는 단계이다. 최종적으로 탐색된 그룹들은 변수 G에 담겨 반환된다.

2) 지속성 판단을 통한 동반 그룹 탐색

이 단계는 매 시간 t_i 마다 탐색된 그룹들을 바탕으로 각 그룹들의 유지 여부, 지속 시간 등을 판단하여 최종적으로 동

```
1:  for each interval
2:    for o_i,o_j in O do
3:      if dist(o_i,o_j) ≤ T_distance
4:        groups_2.add({o_i,o_j})
5:    endFor
6:
7:    for each object o_i contained in groups_2 do
8:      if count(o_i) < T_size −1
9:        remove all groups containing o_i from groups_2
10:   endFor
11:
12:   k ← 2
13:   while () do
14:     for c_i in groups_k do
15:       for c_j in groups_k do
16:         if (c_i[1] = c_j[1]) ∧ (c_i[2] = c_j[2]) ∧ .. ∧ (c_i[k − 1] = c_j[k − 1])
17:           if dist(c_i[k], c_j[k]) ≤ T_distance
18:             groups_{k+1}.add({c_i + c_j[k]})
19:       endFor
20:     endFor
21:     if groups_{k+1} = ∅
22:       break
23:     if k + 1 ≥ T_size
24:       G.add(groups_{k+1})
25:     remove non-maximal groups from G
26:     k ← k + 1
27:   endWhile
28:
29:   return G
30: endFor
```

Fig. 6. Pseudo Code for Periodic Group Detection

반 그룹을 탐색하는 단계이다. 이 단계는 세부적으로 (1) 그룹 지속성 탐색, (2) 중복 그룹 제거, (3) 동반 그룹 반환 세 단계로 구성된다, 아래는 세 단계를 순서대로 설명한다.

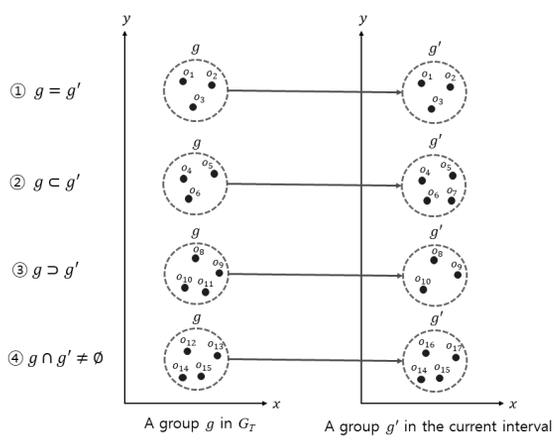

Fig. 7. Four Change Cases of Existing Group

a) 그룹 지속성 탐색

Fig. 7의 ①은 새로 탐색된 그룹과 G_T에 존재하는 기존 그룹이 완전히 동일한 경우이다. 이 경우 Fig. 8의 ①과 같이 G_T에 존재하는 기존 그룹의 정보는 변경되지 않으므로 그대로 유지한다.

Fig. 7의 ②는 새로 탐색된 그룹에 G_T에 존재하는 기존 그룹이 포함되는 경우이다. 이 경우 Fig. 8의 ②와 같이 기존 그룹은 계속 유지되고 있으므로 그대로 유지하고, 새로 탐색된 그룹을 새로운 그룹으로 간주하여 G_T에 새로 추가한다.

Fig. 7의 ③은 새로 탐색된 그룹이 G_T에 존재하는 기존 그룹에 포함되는 경우이다. 이 경우 Fig. 8의 ③와 같이 기존

그룹은 소멸된 것으로 간주하여 G_T에서 제거하고, 새로 탐색된 그룹을 새로운 그룹으로 간주하여 G_T에 새로 추가한다.

Fig. 7의 ④는 새로 탐색된 그룹과 G_T에 존재하는 기존 그룹이 일부 객체를 공유하는 경우이다. 이 경우 Fig. 8의 ④와 같이 기존 그룹과 새로 탐색된 그룹이 공유하는 객체들을 새로운 그룹으로 간주하여 G_T에 추가하고, 기존 그룹은 소멸된 것으로 간주하여 G_T에서 제거한다. 또한 새로 탐색된 그룹도 새로운 그룹으로 간주하여 G_T에 새로 추가한다.

제안 방법은 G_T에 존재하는 모든 그룹과 새로 탐색된 모든 그룹 간에 위 조건을 검사하여 해당하는 작업을 수행한다.

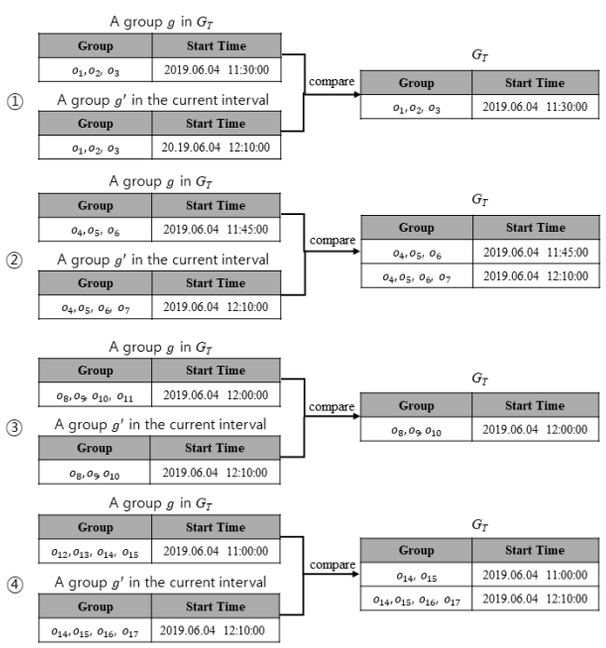

Fig. 8. Examples of Updating Companion Groups Table

b) 중복 그룹 제거

앞 세부단계를 통해 동반 그룹 테이블이 갱신되고 나면 동반 그룹 테이블에는 다양한 그룹들의 정보가 저장된다. 이들 중에는 중복된 그룹이 있을 수도 있고, 다른 그룹에 포함되는 그룹이 있을 수도 있다. 이는 앞 세부단계에서 G_T에 존재하는 모든 그룹들과 새로 탐색된 모든 그룹들 간에 비교를 진행하여 G_T를 갱신하기 때문이다. 따라서 이 단계에서는 동반 그룹 테이블에서 불필요한 그룹들의 정보를 제거한다. 우선 시작 시간이 일치하면서 그룹 객체도 일치하는 경우, 중복되는 그룹 정보를 제거한다. 또한 시작 시간이 일치하면서 다른 그룹에 포함되는 그룹들도 모두 제거한다. 마지막으로 시작 시간이 늦으면서 다른 그룹에 포함되는 그룹들도 모두 제거한다. 이러한 작업을 통해 동반 그룹 테이블에는 동반 그룹 반환에 필요한 그룹들의 정보만 남게 된다.

c) 동반 그룹 반환

중복 그룹 제거 단계를 통해 동반 그룹 테이블이 정리가 되면, 마지막 단계는 사용자로부터 받은 그룹 최소 유지 시간 $T_{duration}$ 이상 유지된 그룹을 찾아 동반 그룹으로 반환하는 단계이다. 이를 위해 동반 그룹 테이블에 저장된 각 그룹에 대해 그의 시작 시간과 현재 시간과의 차이를 구한다. 만약 이 값이 $T_{duration}$ 이상이면 해당 그룹을 동반 그룹으로 반환한다. 또한 그의 생성 시간과 지속 시간을 같이 출력해 준다.

지금까지 설명한 3단계를 매번 그룹 탐색이 수행된 후에 수행하여 동반 그룹을 반환한다. Fig. 9는 본 절에서 설명한 지속성 판단을 통한 동반 그룹 탐색의 의사코드이다. Fig. 9에서 1 - 15번째 줄은 동반 그룹 테이블에 저장되어 있는 그룹들과 현재 시간에 탐색된 그룹들을 비교하여 동반 그룹 테이블을 갱신해주는 단계를 나타낸다. 17 - 18번째 줄은 동반 그룹 테이블로부터 중복된 그룹 또는 다른 그룹에 포함된 그룹들을 제거하는 단계이다. 마지막으로 20 - 24번째 줄은 $T_{duration}$ 이상 지속된 그룹을 동반 그룹으로 반환하는 단계이다.

```
1:  for each group g in G_T do
2:    for each group g' in G do
3:      // G is the set of groups found in the current time interval
4:      if g = g'
5:        do nothing
5:      else if g ⊂ g'
6:        insert (g', the current time) into G_T
7:      else if g ⊃ g'
8:        remove g from G_T
9:        insert (g', the current time) into G_T
10:     else if g ∩ g' ≠ ∅
11:       remove g from G_T
12:       insert (g ∩ g', the current time) into G_T
13:       insert (g', the current time) into G_T
14:   endFor
15: endFor
16:
17: remove duplicate groups from G_T with the same start time
18: remove non-maximal groups from G_T with the same or less start time
19:
20: for each (g, start time) in G_T do
21:   duration time = current time – start time
22:   if (duration time ≥ T_duration)
23:     output (g, start time, duration time)
24: endFor
```

Fig. 9. Pseudo Code for Group Detection with Durability Judgment

5. 성능 평가

본 장에서는 실험을 통해 본 논문에서 제안한 방법의 정확성과 효율성을 평가한다.

5.1 실험환경 및 방법

본 실험에서는 제안 방법의 정확성(correctness)과 성능(performance)을 평가하였다. 본 논문에서는 가상으로 생성한 데이터 스트림으로 실험을 수행하였다. 가상 데이터 스트림은 특정 시작 시간을 기준으로 0.5초씩 증가시키며 생성하였고 모든 객체의 첫 시작은 랜덤으로 좌표 값을 생성하여 객체가 시간에 흐름에 따라 실제로 움직이는 모습을 재현하기 위하여 좌표 값을 랜덤으로 증가 또는 감소시키며 가상 데이

터 스트림을 생성하였다. 예로 들어 좌표 값을 0.1-0.5의 랜덤 값으로 증가 또는 감소시켰다. 또한 그중에 몇몇 객체들은 특정 시간에는 생성되고 사라지는 모습을 재현하였다. 즉, 객체가 시간의 흐름에 따라 자연스럽게 움직이는 모습을 재현하였다. 정확성 실험은 데이터 스트림에 대해 제안 방법이 동반 그룹을 올바르게 찾느냐를 확인하였다. 성능 실험은 역시 데이터 스트림의 레코드를 증가시켜가며 제안 방법으로 모든 레코드들을 처리하는 데 걸린 총 누적 시간을 측정하였다. (주기적 그룹 탐색 단계 수행 시간 + 지속성 판단을 통한 동반 그룹 탐색 단계 수행 시간) 앞서 언급한 바와 같이 현재 동반 그룹을 탐색함에 있어 거리를 기반으로 그룹을 탐색하는 연구는 이루어진 바가 없다. 따라서 본 논문에서는 비교 대상으로는 본 논문에서 정의한 4.1에서 설명한 단순 방법으로 주기적으로 그룹을 탐색하는 방법과 비교하였다. 제안 방법과 단순 방법 모두 Python으로 구현하였다. 실험은 운영체제로 윈도우 10을 사용하며, CPU는 Intel i5-7400K이고 메모리는 24 GB인 데스크탑 컴퓨터에서 수행하였다.

5.2 실험 결과

첫 번째 실험은 제안 방법의 정확성을 실험하였다. 데이터 스트림의 발생 시간은 총 100분, 시간 간격 $\triangle t = 10$분으로 하였으며, 객체 수는 40개로 동반 그룹은 6개를 이루는 가상 데이터를 생성하여 사용 하였다. 본 실험은 그 결과를 눈으로 직접 확인하기 위해 비교적 적은 양의 데이터 스트림을 대상으로 실행하였다. 사용된 가상 데이터는 동반 그룹 6개 이외에는 동반 그룹이 나타나지 않게 생성되었다. 사용자 매개변수는 $T_{distance}$=2, T_{size}=3, $T_{duration}$=30분으로 하였다. Fig. 10은 데이터 스트림에 포함된 모든 데이터들의 각 시간 별 좌표와 제안 방법이 탐색한 6개의 동반 그룹을 나타낸다. Fig. 10에서 보이는 바와 같이 제안 방법은 데이터 스트림에 나타나는 모든 객체들 중에서 서로 거리가 2 이내이며, 객체 수가 3개 이상이고, 지속 시간이 30분 이상인 동반 그룹 6개를 모두 정확히 탐색함을 확인할 수 있었다.

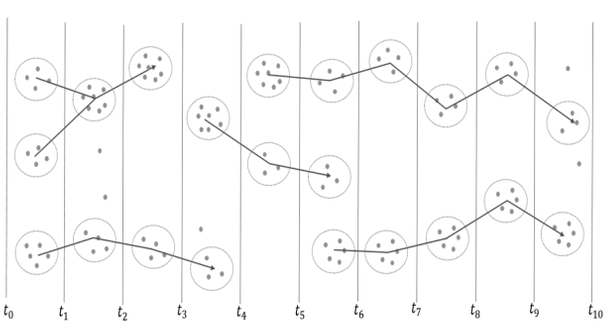

Fig. 10. Correctness Experiment Results

두 번째 실험은 데이터 스트림에 나타나는 객체들의 수를 증가시켜가며 제안 방법과 단순 방법으로 동반 그룹을 탐색하는 데 걸리는 총 누적 시간을 비교하였다. 객체의 수는 2000, 3000, 4000, 5000으로 증가시켰으며, 3시간 동안 데이터 스트림을 발생시켰고, 10분 간격으로 그룹 탐색을 수행하였다. 단순 방법과 제안 방법 모두 사용자 임계값 $T_{distance}$=6, T_{size}=10, $T_{duration}$=20으로 고정시켜 실험을 진행하였다. Fig. 11은 객체 수 증가에 따른 성능 실험의 결과를 나타낸다. 객체의 수가 커지면 커질수록 두 방법 모두 동반 그룹을 탐색하는 데 걸리는 총 수행 시간이 증가한다. 하지만 제안 방법은 단순 방법에 비해 좋은 성능을 보이고 있다. 객체의 수가 증가할수록 두 방법 모두 2개의 객체로 이루어진 그룹을 찾는 시간과 해당 그룹들을 확장하는 데 드는 비용이 증가한다. 하지만 제안 방법은 기존 방법과 달리 $k+1$개의 객체로 이루어진 그룹을 생성하기 위해 k개의 객체로 이루어진 그룹들 중 $k-1$개의 객체가 일치하는 그룹들 간에만 병합을 고려한다. 따라서 그룹을 확장시킬 때 발생하는 불필요한 비교 및 거리 계산 비용이 크게 감소된다.

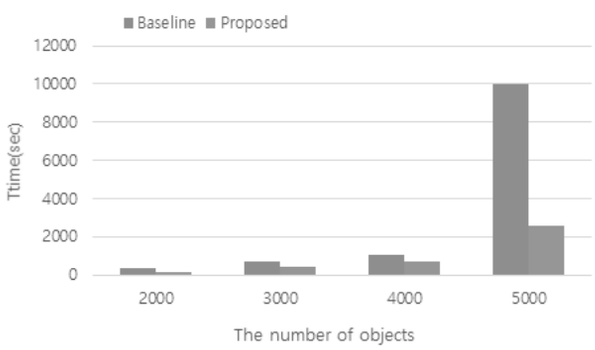

Fig. 11. Performance Evaluation Results with Increasing Number of Objects

세 번째 실험은 모든 조건이 동일할 때 $T_{distance}$ 값을 변화시켜가며 제안 방법과 단순 방법의 성능을 비교하였다. 객체의 수는 150개로 하였으며, 3시간 동안 데이터 스트림을 발생시켰고, 10분 간격으로 그룹 탐색을 수행하였다. $T_{distance}$는 5, 10, 15, 20으로 증가시켰다. 단순 방법과 제안 방법 모두 사용자 임계값을 T_{size}=5, $T_{duration}$=30으로 고정시켜 비교 실험을 진행하였다. Fig. 12은 거리 임계값에 따른 성능 평가 실험의 결과를 나타낸다. $T_{distance}$가 커지면 커질수록 두 방법 모두 총 수행 시간이 증가함을 관찰할 수 있다. 이는 $T_{distance}$가 커질수록 그룹을 구성하는 객체 수가 많아지며, 이에 따라 그룹 확장이 더 많이 진행되어야 하기 때문이다. 하지만 두 번째 실험과 마찬가지로 제안 방법은 단순 방법에 비해 그룹 확장에 드는 비용이 매우 적기 때문에 전체 수행 시간이 크게 감소됨을 볼 수 있다.

네 번째 실험은 모든 조건이 동일할 때 T_{size} 값을 변화시켜가며 제안 방법과 단순 방법의 성능을 비교하였다. 객체의 수는 150개로 하였으며, 3시간 동안 데이터 스트림을 발생시켰고, 10분 간격으로 그룹 탐색을 수행하였다. T_{size}는 5, 10, 15, 20으로 증가시켰다. 단순 방법과 제안 방법 모두 사용자

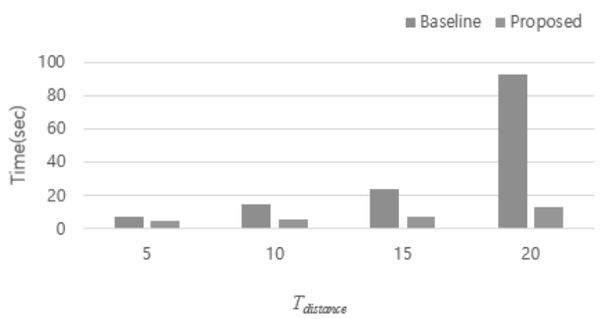

Fig. 12. Performance Evaluation According to Tdistance Value

임계값을 $T_{distance}$=5, $T_{duration}$=30으로 고정시켜 비교 실험을 진행하였다. Fig. 13은 거리 임계값에 따른 성능 평가 실험의 결과를 나타낸다. T_{size}가 커지면 커질수록 두 방법 모두 수행 시간이 감소함을 관찰할 수 있다. 이는 T_{size}가 커질수록 조건을 만족하는 그룹 수가 감소하여 동반 그룹 테이블 G_T에 저장된 그룹들의 수가 줄어들며, 그에 따라 지속성 판단을 통한 동반 그룹 탐색 단계의 수행 시간이 감소하기 때문이다. 반면 그룹 탐색 단계의 수행 시간은 T_{size}에 크게 영향을 받지 않는다. 하지만 제안 방법이 그룹을 탐색하는데 걸리는 시간은 단순 방법보다 훨씬 적기 때문에 결국 제안 방법의 총 수행 시간은 단순 방법에 비해 크게 줄어든다.

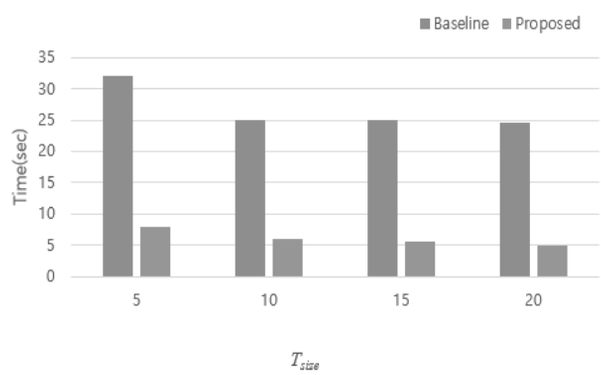

Fig. 13. Performance Evaluation According to Tsize Value

마지막 실험은 모든 조건이 동일할 때 $T_{duration}$ 값을 변화시켜가며 제안 방법과 단순 방법의 성능을 비교하였다. 객체의 수는 150개로 하였으며, 3시간 동안 데이터 스트림을 발생시켰고, 10분 간격으로 그룹 탐색을 수행하였다. $T_{duration}$는 5, 10, 15, 20으로 증가시켰다. 단순 방법과 제안 방법 모두 사용자 임계값을 $T_{distance}$=5, T_{size}=5 로 고정시켜 비교 실험을 진행하였다. Fig. 14은 누적 시간 임계값에 따른 성능 평가 실험의 결과를 나타낸다. $T_{duration}$가 증가해도 두 방법 모두 수행 시간의 변화가 거의 없는것을 관찰할 수 있다. 이는 $T_{duration}$ 값은 크게 주더라도 동반 그룹 탐색 단계의 수행 시간에는 영향을 미치지 않는 다는 것을 의미한다.

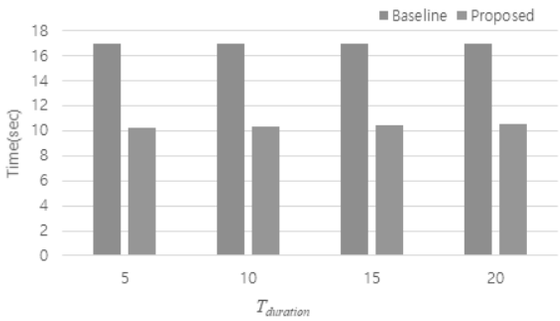

Fig. 14. Performance Evaluation According to Tduration Value

6. 결 론

본 논문은 궤적 데이터 스트림에서 동반 그룹을 탐색하는 새로운 방법을 제안하였다. 지금까지 기존 연구들은 특정 시점에 서로 가까이 존재하는 객체들의 그룹들을 탐색하기 위해 기존의 클러스터링 기법을 사용하였다. 하지만 이들은 클러스터링에 필요한 매개변수를 지정하기 어렵거나 클러스터의 모양 및 크기를 제어하기 어렵다는 단점이 있다. 이에 비해 제안 방법은 사용자가 지정한 객체 간 최대 거리 $T_{distance}$를 기반으로 동반 그룹을 탐색한다. 따라서 제안 방법은 사용자가 원하는 동반 그룹을 명확히 지정할 수 있다. 제안 방법은 서로 $T_{distance}$ 내에 존재하는 객체들의 그룹을 효율적으로 찾기 위해 기존의 연관 규칙 탐색에 널리 사용되는 Apriori 알고리즘과 비슷한 전략을 사용한다. 제안 방법은 동반 그룹이 될 가능성이 없는 불필요한 그룹 탐색을 피함으로써 매우 효율적으로 서로 $T_{distance}$ 내에 존재하는 객체들의 그룹을 탐색할 수 있다. 또한 제안 방법은 기존 방법과 달리 동반 그룹을 구성하는 객체 외에도 그의 생성 시간과 지속 시간을 출력해준다. 앞서 설명한 바와 같이 지금까지 궤적 데이터 스트림에서 동반 그룹을 탐색하기 위하여 모두 클러스터링을 기반으로 탐색하였고 거리 기반으로 탐색하는 연구는 이루어진 바가 없기 때문에 본 논문은 거리를 기반으로 하는 가장 단순한 방법과 제안 방법을 비교 실험 하였다. 실험을 통해 제안 방법의 정확성을 보이는 한편, 객체의 수 T_{size}, 객체 간의 거리 $T_{distance}$ 등을 변화시키는 다양한 실험을 통해 모든 경우에 제안 방법이 단순 방법에 비해 매우 효율적으로 동반 그룹을 탐색함을 보였다.

References

[1] J.-G. Lee, J. Han, and K.-Y. Whang, "Trajectory Clustering: a Partition-and-group Framework," in *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD '07)*, pp. 593-604, ACM, Beijing, China, June 2007.

[2] P. Bakalov, M. Hadjieleftheriou, and V. J. Tsotras, "Time Relaxed Spatiotemporal Trajectory Joins," *Proceedings of*

the 13th Annual ACM International Workshop on Geographic Information Systems, Nov. 04-05, 2005.

[3] S. Atev, O. Masoud, and N. Papanikolopoulos, "Learning Traffic Patterns at Intersections by Spectral Clustering of Motion Trajectories," in *Proc. IEEE Conf. Intell. Robots Syst.*, Bejing, China, Oct. pp.4851-4856, 2006.

[4] L. Chen, M. T. Ozsu, and V. Oria, "Robust and Fast Similarity Search for Moving Object Trajectories," in *Proc. ACM SIGMOD Int'l Conf. Management of Data*. New York, NY, USA: ACM Press, pp.491-502, 2005.

[5] P. Kalnis, N. Mamoulis, and S. Bakiras. "On Discovering Moving Clusters in Spatio-temporal Data," In *SSTD*, pp.364-381, 2005.

[6] J. Gudmundsson and M. van Kreveld, "Computing Longest Duration Flocks in Trajectory Data," In *ACM GIS*, pp.35-42, 2006.

[7] Z. Li, B. Ding, J. Han and R. Kays, "Swarm: Mining Relaxed Temporal Moving Object Clusters," *Proceedings of the VLDB Endowment 3*, 1-2, 723-734, 2010.

[8] J. Gudmundsson, M. V. Kreveld, and B. Speckmann, "Efficient detection of motion patterns in spatiotemporal data sets," In *the Proceedings of the 12th International Conference on Advances in Geographical Information Systems*, ACM, 250-257, 2004.

[9] J, Mao and Q. Song, "Online Clustering of Streaming Trajectories," Frontiers of Computer Science, Volume 12, Issue 2, pp.245-263, 2018

[10] Z. Li, J. Lee, X. Li, and J. Han, "Incremental Clustering for Trajectories," *Proc. Database Systems Advanced Applications*, pp.32-46, 2010.

[11] T. L. C. Da Silva, K. Zeitouni, and J. A. F. De Macedo, "Online Clustering of Trajectory Data Stream," in *Proceedings of the 17th IEEE International Conference on Mobile Data Management (IEEE MDM '16)*, pp.112-121, Jun. 2016.

[12] Y. Yanwei, Q. Wang, X. Wang, H. Wang, and J. He, "Online Clustering for Trajectory Data Stream of Moving Objects," *Computer Science and Information Systems*, Vol.10, No.3, pp.1293-1317, 2013.

[13] S. Tasnim, "Semantic-Aware Clustering-based Approach of Trajectory Data Stream Mining," ICNC, 2018.

[14] Riyadh, Musaab, Norwati Mustapha, Md Sulaiman and Nurfadhlina Binti Mohd Sharef. "CC_TRS: Continuous Clustering of Trajectory Stream Data Based on Micro Cluster Life," Mathematical Problems in Engineering 2017.

[15] S. Elnekave, M. Last and O. Maimon, "Incremental Clustering of Mobile Objects," in *Proceedings of the Workshops in Conjunction with the 23rd International Conference on Data Engineering (ICDE '07)*, pp.585-592, Apr. 2007.

[16] L. Tang et al., "On Discovery of Traveling Companions from Streaming Trajectories," *Proc. ICDE*, pp.186-197, 2012.

[17] T. T. Shein, S. Puntheeranurak, and M. Imamura, "Incremental Discovery of Crowd from Evolving Trajectory Data," in *2018 International Conference on Engineering, Applied Sciences, and Technology (ICEAST)*, 2018.

[18] S. Puntheeranurak, T. T. Shein, and M. Imamura, "Efficient Discovery of Traveling Companion from Evolving Trajectory Data Stream," in *2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC)*, Vol. 01, pp.448-453, Jul. 2018.

[19] R. Agrawal and R. Srikant, "Fast Algorithms for Mining Association Rules," In VLDB'94, pp.487-499, 1994.

[20] A. Ceglar and J. F. Roddick, "Association mining," *ACM Computing Surveys (CSUR)*, Vol.38, No.2, 2006.

강 수 현

https://orcid.org/0000-0003-2606-6836
e-mail : suhyunkang@sookmyung.ac.kr
2016년 평택대학교 정보통신학과(학사)
2019년 숙명여자대학교 컴퓨터과학과 석사
관심분야 : 데이터베이스, 데이터마이닝, 빅데이터, 데이터스트림

이 기 용

https://orcid.org/0000-0003-2318-671X
e-mail : kiyonglee@sookmyung.ac.kr
1998년 KAIST 전산학과(학사)
2000년 KAIST 전산학과(석사)
2006년 KAIST 전산학과(박사)
2006년~2008년 삼성전자 책임연구원
2008년 ~ 2010년 KAIST 전산학과 연구조교수
2010년 ~ 현 재 숙명여자대학교 소프트웨어학부 교수
관심분야 : 데이터베이스, 데이터마이닝, 빅데이터, 데이터스트림