공간데이타 마이닝을 위한 효율적인 그리드 셀 기반 공간 클러스터링 알고리즘

문 상 호[†] · 이 동 규[††] · 서 영 덕[†††]

요 약

대용량의 공간데이터베이스로부터 암시적이고 유용한 지식을 자동적으로 추출하는 공간데이터 마이닝은 데이타 양이 급격히 증가하면서 필요성이 더욱 중대되고 있다. 공간데이터 마이닝에서 데이타를 분석하여 유사한 그룹으로 분류하는 공간 클러스터링은 매우 중요한 분야이다. 기존 연구에서 공간 클러스터링을 위한 여러 가지 알고리즘들이 제시되었지만, 다음과 같은 문제점들이 있다. 먼저 클러스터링을 위하여 객체들 간의 거리를 기반으로 하므로 데이타 양이 많아질수록 계산 비용이 커진다. 또한, 메모리 상주 데이타를 대상으로 하므로 대용량의 데이타인 경우에 효율이 떨어진다. 본 논문에서는 공간데이터 마이닝을 위하여 그리드 셀을 기반으로 한 효율적인 공간 클러스터링 방법을 제시한다. 이 클러스터링에서는 기존 공간 클러스터링 기법들의 문제점을 해결하는데 중점을 둔다. 세부적으로 공간 클러스터링의 효율성을 높이기 위하여 클러스터링시에 발생하는 비용(계산량)을 감소시키는 것이다. 이를 위해서 공간지역성을 보장하는 대표적인 공간분할 방법인 그리드 셀을 기반으로 한 공간 클러스터링 기법을 제시한다.

An Efficient Grid Cell Based Spatial Clustering Algorithm for Spatial Data Mining

Sang-Ho Moon[†] · Dong-Gyu Lee[††] · Young-Duck Seo[†††]

ABSTRACT

Spatial data mining, i.e., discovery of interesting characteristics and patterns that may implicitly exists in spatial databases, is a challenging task due to the huge amounts of spatial data. Clustering algorithms are attractive for the task of class identification in spatial databases. Several methods for spatial clustering have been presented in recent years, but have the following several drawbacks : increase costs due to computing distance among objects and process only memory-resident data. In this paper, we propose an efficient grid cell based spatial clustering method for spatial data mining. It focuses on resolving disadvantages of existing clustering algorithms. In details, it aims to reduce cost further for good efficiency on large databases. To do this, we devise a spatial clustering algorithm based on grid cell structures including cell relationships.

키워드 : 공간 클러스터링(Spatial Clustering), 클러스터링 알고리즘(Clustering Algorithm), 공간데이터 마이닝(Spatial Data Mining)

1. 서 론

현대 사회는 정보 기술의 급속한 발달로 인하여 데이타가 폭발적으로 증가하고 있고, 데이타의 무제한적인 증가는 원하는 정보의 접근을 점차적으로 불가피하게 어려운 상황으로 만들고 있다[5, 6, 12]. 이로 인하여 급속도로 증가하는 데이타로부터 존재하지만 분석되지 않는 유용한 지식의 탐사에 대한 연구가 활발하게 진행되고 있다. 지식탐사는 대용량 데이타베이스로부터 암시적이며 이전에 알려지지 않은 잠재적으로 유용한 지식을 추출하는 과정이다[1-4, 8-10]. 일반적으로 지식탐사는 데이타 선정, 정제(cleaning), 보강(enrichment), 코딩, 데이타 마이닝, 보고서 작성의 단계를 거친다[12].

대부분의 데이타 접근은 SQL이나 통계 도구만을 사용하더라도 원하는 데이타를 얻을 수 있고, 일부는 데이타 마이닝을 사용하지 않고도 적절한 정보를 얻을 수 없는 경우도 있다. 데이타 마이닝의 필수적인 이유는 숨겨져 있어서 접근할 수 없는 20%의 데이타가 잠재적으로 매우 중요한 의미를 가지기 때문이다[5, 12]. 현재 데이타 마이닝은 이론적 시도에서 벗어나 그 활용 범위가 마케팅, 시장 분석, 계획, 스케줄링 등과 같이 다양한 분야에 널리 이용되고 있다. 그

† 정 회 원 : 부산외국어대학교 컴퓨터공학부 교수
†† 정 회 원 : 에이츠컴 자동화사업부/제어팀 연구원
††† 정 회 원 : 부산대학교 대학원 컴퓨터공학과
논문접수 : 2002년 6월 5일, 심사완료 : 2003년 3월 31일

리고 연관 규칙, 클러스터링, 분류 등의 다양한 데이타 마이닝 기법을 이용하여 지식을 얻을 수 있다. 데이타 마이닝 기법들 중에서 클러스터링은 데이타 공간의 데이타 점들에서 유사한 특징을 가진 점들을 집단화하는데 매우 유용한 기법이며, 클러스터 분석은 얻어진 데이타의 분포로부터 유용한 지식을 얻을 수 있다[1-3, 7, 11-13]. 따라서 이러한 클러스터링 기법은 유사성 검색, 고객 분류, 경향 분석 등을 위한 방법론으로 활발하게 연구되고 있는 실정이다.

공간데이터 마이닝은 일반적인 마이닝 기법을 공간 도메인에 적용한 경우로, 공간 DB로부터 암시적이며 잠재적인 지식을 추출하는 과정을 의미한다[1-3, 8]. 공간 클러스터링은 공간데이터 마이닝 기법중의 하나로, 공간객체들에 대하여 공간적 특성을 이용하여 집단화하는 과정이다. 이러한 공간 클러스터링은 일반적인 클러스터링과 마찬가지로 중요한 공간데이터 마이닝 방법으로, 다른 알고리즘의 전처리 단계로 이용되거나 유사성 검색 등의 많은 응용 분야에 널리 사용되고 있다.

데이타 마이닝 작업에서 사용되는 데이타의 크기는 그 특성상 대규모를 이루고 있다. 특히, 공간 DB는 일반 DB와는 달리 속성데이타 이외에 다양하고 복잡한 공간데이터를 포함하기 때문에 데이타 양이 방대하다. 이러한 공간데이터의 특성으로 인하여 일반 데이타에 비하여 탐색공간의 복잡도가 커진다. 그리고 복잡하고 대용량인 공간데이터를 대상으로 하여 암시적이고 잠재적인 유용한 지식을 발견하기 위해서는 많은 비용이 든다. 그러므로 효율적인 공간데이터 마이닝을 위해서 전체 데이타로부터 의미 있는 부분집합(클러스터)을 발견하고, 발견된 데이타의 부분집합을 대상으로 유용한 지식을 추출하는 것이 바람직하다. 따라서 공간데이터 마이닝에서 공간 클러스터링은 중요한 역할을 담당하며, 복잡도가 큰 탐색 공간에서 효율적인 공간 클러스터링 알고리즘의 제시가 필요하다.

기존 연구에서 공간 클러스터링을 위한 많은 알고리즘들이 제시되었지만, 다음과 같은 문제점들이 있다. 먼저 대부분의 알고리즘들은 클러스터링을 위하여 객체들간의 거리를 계산하므로 데이타 양이 많아질수록 계산 비용이 커진다. 또한, 메모리 상주 데이타를 대상으로 하므로 대용량의 데이타인 경우에 효율이 떨어진다. 본 논문에서 제시하고자 하는 공간 클러스터링 기법은 기존 클러스터링 기법들의 단점을 극복하는데 주안점을 두고자 한다. 특히, 방대한 양의 공간데이터에 대한 효율적인 클러스터링을 위한 비용(계산량) 감소에 중점을 둔다. 따라서 본 논문에서는 공간데이터의 특성으로 인한 탐색공간의 복잡도를 효율적으로 처리할 수 있는 클러스터링 기법 제시가 주된 목적이다. 이를 위해서 대표적인 공간분할 방법인 그리드 셀을 기반으로 한 공간 클러스터링 기법을 제시한다.

2. 관련 연구

DBSCAN은 공간데이터 마이닝을 위해 밀도를 기반으로 한 클러스터링 알고리즘이다[2]. 이 알고리즘에서 중요한 매개변수로 *Eps*와 *MinPts*를 이용한다. *MinPts*는 클러스터에 포함되는 최소 점들의 수를 나타내며, *Eps*는 밀도를 나타낸다. 세부적으로 공간 DB의 각 점에 대하여 4번째(k = 4) 가까운 점들 사이의 거리를 계산한 후에, 이 거리에 따라 점들을 정렬하여 k-dist 그래프를 그린다. 이 그래프는 밀도 *Eps*를 구하는데 이용하는 것으로, 그래프 상에서 잡음과 클러스터를 구분하는 경계를 찾은 후에 이 경계값을 *Eps*로 정한다. 실제로 *Eps* 값을 기준으로 공간 DB상의 모든 점들에 대하여 거리를 계산한 후에, *Eps* 값 내에 있는 점들에 대하여 클러스터링을 한다. DBSCAN의 특징은 공간데이터 마이닝을 위하여 클러스터링에서 적용되는 매개변수를 실제 데이터의 밀도에 근거한 값(*Eps*)을 구한다는 것이다. 그러나 k-dist 그래프를 생성하기 위하여 각 점들에 대하여 거리를 계산해야 하고, 그래프 상에 경계값을 정하기 위한 사용자 분석이 필요하다. 또한, 실제 클러스터링을 위하여 각 점들에 대하여 거리를 구한 후에 *Eps* 값과 비교해야 한다. 따라서 DBSCAN은 *Eps* 값 산출과 클러스터링을 위해 공간의 모든 점들에 대한 거리를 구하기 위한 계산량이 많아진다는 단점이 있다.

CLARANS는 기존의 데이타 마이닝을 위해 제안된 PAM과 CLARA를 결합하여 제안한 알고리즘이다[1]. 이 방법에서는 데이타 집합의 부분집합만을 검색하며, 검색의 각 단계에서 무작위로 샘플을 생성하여 이용한다. 공간 DB 상에서 하나의 점을 *medoid*로 대체한 후에 이 점을 대상으로 클러스터링 과정을 수행한다. 클러스터링 과정에서 더 좋은 이웃이 있으면, *medoid*를 옮겨서 클러스터링 과정을 다시 수행한다. 그렇지 않으면 현재 클러스터링에서 생성된 클러스터를 결정한다. 그리고 하나의 클러스터 생성이 완료되면, 새로운 클러스터를 생성하기 위하여 무작위로 선정된 새로운 점을 *medoid*로 선정하여 클러스터링 과정을 수행한다. 이 알고리즘에서 중요한 변수로는 이웃들의 수를 제한하는 *maxneighbour*, 검색된 클러스터의 수를 제한하는 *numlocal* 등이 있다. CLARANS는 기존의 마이닝 기법을 공간데이터 마이닝에 적용한 최초의 공간 클러스터링 알고리즘이지만, 클러스터링 과정에서 기본적으로 점들간의 거리를 이용하므로 계산량이 많아진다.

H-SCAN은 공간데이터를 클러스터링하기 위하여 1차원을 위한 해시 방법을 확장하여 d 차원 공간에서 사용한 것이다[8]. 이 알고리즘은 두 단계로 수행된다. 첫 번째 단계에서는 공간 DB로부터 읽은 객체들을 사용하여 공간 해시 구조를 생성한다. 세부적으로 전체 공간 도메인은 공간 해시함수를 통해 작은 셀들로 나누어지고, 각 셀에 소속한 객

체들의 수와 객체에 대한 참조포인터를 갖는다. 두 번째 단계에서는 해시구조를 기반으로 사용자의 요구에 따라 클러스터링한다. 클러스터링 과정은 해시구조를 이용하여 객체의 수가 임계치 이상인 셀들 중에서 서로 연결된 셀들의 집합을 클러스터로 생성한다. H-SCAN은 기존의 공간 클러스터링이 주기억장치를 이용하는 단점을 해결하기 위하여, 공간 해시구조를 기반으로 디스크를 사용하여 데이타를 저장한다. 그러나 이 방법에서는 셀 크기와 셀에 속한 객체의 수에 대한 기준 제시가 되지 않았다. 다만, 밀도에 따라 셀 크기를 결정한다는 일반적인 사항만 언급하고 있고, 셀 내에 포함된 객체들의 수를 기준으로 클러스터를 생성하므로 셀 크기에 따라 클러스터링 결과가 달라진다. 따라서 클러스터 생성에 있어서 임계 값을 반영하기가 어렵다.

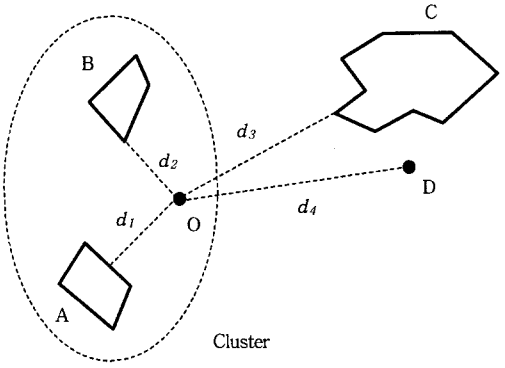
STING 알고리즘은 공간데이터 마이닝을 위하여 통계정보를 그리드 셀 계층구조를 관리하여 이용한다[3]. 먼저 전체 공간을 일정한 크기의 셀로 분할하여 상향식으로 4개의 셀들을 묶어서 *i*, *i-1*, ···, *1*(*root*) 레이어를 생성한 후에, 전체적으로 다중 레벨의 레이어들로 구성되는 계층구조를 형성한다. 그리고 단말노드의 셀에는 속성-비의존과 속성-의존 변수값을 가진다. 전자는 셀에 속한 객체의 수(*n*)가, 후자는 셀 내에 있는 객체들의 숫자값을 가지는 속성으로 평균(*m*), 표준편차(*s*), 최소값(*min*), 최대값(*max*), 분포(*dist*) 값이 해당된다. 셀 계층구조를 생성 할 때 먼저 단말노드의 셀들을 만들고, 각 셀 내에 있는 객체들로부터 *n*, *m*, *s*, *min*, *max*, *dist*와 같은 통계 정보를 계산한 후에 저장한다. 그리고 단말노드의 셀들로부터 구성되는 상단부의 레이어들은 하단부 레이어의 4개 셀들로 구성되므로, 이 셀에 저장되어 통계 정보들로부터 현재 셀의 *n*, *m*, *s*, *min*, *max*, *dist* 값들을 구한다. 이와 같은 작업을 최상위 루트 레이어에 있는 셀까지 반복하여 수행한다.

이 알고리즘은 셀 계층구조에 있는 통계정보를 이용하여 공간데이터 마이닝과 연관된 영역질의를 수행할 때 효율적이다. 예를 들어, "단위 영역당 주택이 100가구 이상이고 주택의 70% 이상이 400,000달러인 최대 영역을 구하라"와 같은 질의인 경우에는 전체 DB를 검색하지 않고, 셀 계층구조를 하향식으로 검색하면서 적정한 레이어를 찾아서 이 레이어의 통계정보를 이용하여 값을 얻을 수 있다. 본 논문에서 제시하는 알고리즘과의 차이는 STING은 셀 계층구조를 이용하여 통계정보를 관리 및 저장하고, 이 정보를 향후 영역질의에 이용한다. 즉, 단순히 통계정보를 관리하기 위하여 셀들 간의 계층구조를 이용한다는 것이다.

3. 공간 클러스터링 기법

3.1 기존 클러스터링

기존 연구에서 제시된 대부분의 클러스터링 기법은 객체간의 거리를 기반으로 한다. 즉, 기존의 클러스터링 과정은 먼저 각 객체들 간의 거리를 계산하여, 임계값(기준값)보다 작으면 클러스터에 포함시킨다. 예를 들어 (그림 1)에서 객체 *O*를 기준점으로 하여 객체들 간의 거리를 구하면, 객체 *A*, *B*와의 거리 d_1, d_2는 임계값 내에 있으므로 클러스터에 포함된다. 그러나 객체 *C*, *D*와의 거리 d_3, d_4는 임계값보다 크므로 클러스터에 포함되지 않는다.



(그림 1) 기존 클러스터링 과정

기존 클러스터링의 문제점은 객체들간의 거리 연산에 많은 비용이 든다는 것이다. 즉, 기준 객체와 다른 객체들 간에 거리 계산이 필요하고, 다른 클러스터를 찾는 과정에서도 이 과정이 반복되므로 많은 비용이 든다. 특히, 공간 클러스터링인 경우에는 점 객체 이외에 선, 다각형 객체들 간의 거리를 계산해야 하므로 비용이 급격하게 증가한다.

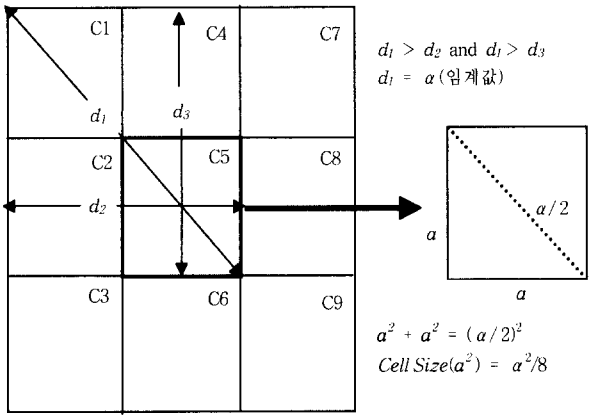
3.2 그리드 셀 기반 클러스터링

본 논문에서는 기존 클러스터링의 문제점을 극복하기 위하여 공간지역성을 보장하는 대표적인 공간분할 방법인 그리드 셀을 구조를 이용한 공간 클러스터링 기법을 제시한다. 이 기법의 핵심 아이디어는 그리드 셀 구조를 기반으로 한 셀 관계 연산을 통하여 클러스터링을 하는 것이다. 이것은 객체들간의 거리 계산 대신에 셀들간의 관계를 이용하므로, 기존 클러스터링 과정에서 많은 비용을 차지하는 거리 계산을 최소화 할 수 있다. 따라서 기존 클러스터링 방법에서 거리 계산에 의한 비용을 줄일 수 있는 장점이 있다.

이 클러스터링에서는 먼저 전체 공간 영역에 대하여 그리드 셀 구조를 생성해야 한다. 이 셀 구조의 생성에서 가장 중요한 것은 셀의 크기를 결정하는 것이다. 즉, 거리 계산없이 셀 관계 연산만으로 클러스터링이 가능하게 전체 공간 영역을 일정한 크기의 셀들로 나누어야 한다. 여기서는 클러스터링을 위하여 사용자가 제시하는 임계값을 기준으로 하여 셀 크기를 결정한다.

(그림 2)는 사용자가 부여한 임계값(α)을 기준으로 셀 크기를 결정하는 것을 보여준다. *C5*셀은 *C1* 등을 포함한 8

개의 셀들과 인접하고, 이 셀을 기준으로 인접 셀들과 가장 먼 거리는 대각선으로 인접하는 셀과의 대각선 거리이다. 즉, *C5*와 *C1*과의 대각선 거리(d_1)는 *C5*와 수평으로 인접한 *C2*와의 거리(d_2) 또는 수직으로 인접한 *C4*와의 거리(d_3)보다 크다. 본 논문에서는 *C5*셀을 기준으로 가장 먼 거리인 d_1을 임계값과 동일한 값으로 길이를 정한다. 이렇게 하면 *C5*셀에 있는 객체들은 인접한 *C1* 등 8개의 셀들에 포함된 객체들과는 거리 계산없이 하나의 클러스터로 구성할 수 있다. 이유는 *C5*셀과 인접 셀들간에 가장 먼 거리가 $d_1(d_1 = \alpha)$이므로, *C5*셀과 인접셀 내에 있는 모든 객체들은 임계값 이내에 있기 때문이다.



(그림 2) 셀 크기 결정

대각선으로 인접한 셀인 경우에 셀들간의 가장 먼 거리가 대각선들의 합(d_1)이므로 셀 내의 모든 객체들은 임계값 내에 있으므로, *C5*셀과 *C1, C3, C7, C9*셀 내에 포함된 객체들은 하나의 클러스터로 묶을 수 있다. 수평 또는 수직으로 인접한 셀인 경우에는 셀들간의 가장 먼 거리가 수평 또는 수직 길이의 합(d_2 또는 d_3)이므로 임계값보다 작다(d_2 or $d_3 < d1$). 따라서 *C5* 셀과 *C2, C4, C6, C8*셀 내에 포함된 객체들도 임계값 내에 있으므로 클러스터로 구성할 수 있다. 결론적으로 *C5*셀을 기준으로 인접한 8개의 셀인 *C1, C2, C3, C4, C6, C7, C8, C9*는 셀 관계를 통하여 하나의 클러스터를 형성하게 된다.

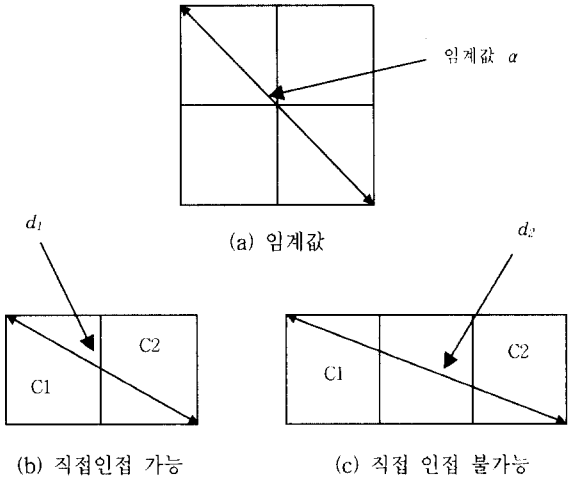
4. 공간 클러스터링 알고리즘

4.1 셀 관련성 정의

본 절에서는 먼저 공간 클러스터링 알고리즘에서 이용되는 셀들간의 관련성들을 정의한다. 이러한 정의들은 기존의 클러스터링 방법에서 사용되었던 객체들간의 거리 계산을 대체하기 위한 그리드 셀들간의 관계 연산으로 이용된다.

직접 인접(Direct Adjacent) : 그리드 셀 구조에서 임의의 셀 *C1*과 *C2*간의 최대거리가 임계값보다 작거나 같으면, 이 두 셀 들은 직접 인접한다.

(그림 3)은 셀들 간의 직접 인접의 예를 보여준다. 먼저 (그림 3)(a)에서는 임계값을 기준으로 셀 크기를 결정한 것을 보여준다((그림 2) 참조). (그림 3)(b)에서 *C1, C2* 셀간의 최대거리(d_1)는 임계값보다 작으므로($d_1 < \alpha$), 이 두 셀 들은 직접인접 관계를 가진다. 반면에 (그림 3)(c)의 *C1, C2* 셀간의 최대거리(d_2)는 임계값보다 크므로($d_2 > \alpha$) 직접 인접하지 않다.



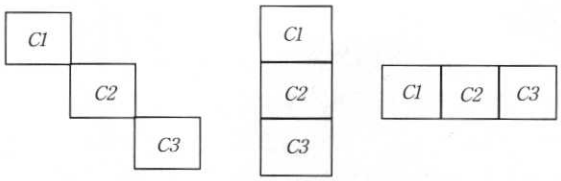
(그림 3) 직접인접 예

직접 인접 셀(DAC : Direct Adjacent Cells) : 그리드 셀 구조에서 기준 셀에 대하여 직접인접 관계에 있는 셀들로 정의한다. 그리드 셀 구조상에서 임의의 셀에 대하여 직접 인접 셀은 최소 3개에서 최대 8개이다.

(그림 2)에서 *C5*셀을 기준으로 하면 직접인접 셀은 *C1, C2, C3, C4, C6, C7, C8, C9*가 되고, *C7*셀을 기준으로 하면 *C4, C5, C8*이 된다. 이러한 직접인접 셀은 공간 클러스터링 알고리즘에서 중요하게 이용되는 정의이다. 즉, 클러스터 생성 알고리즘에서 임의의 셀에 직접인접 셀들은 거리 계산없이 셀들 내의 모든 객체들을 하나의 클러스터로 생성할 수 있다.

인접 가능(Enable Adjacent) : 그리드 셀 구조에서 임의의 셀들 *C1, C2, C3*에 대하여, *C1*과 *C2*가 직접인접하고 *C2*와 *C3*가 직접인접하면 *C1*과 *C3*는 인접 가능하다.

(그림 4)는 그리드 셀 구조에서 인접 가능한 경우를 보여준다. 여기서 *C2*셀을 기준으로 하여 *C1*과 *C3*셀이 각각 대각선, 수직, 수평으로 인접 가능한 경우의 예이다. 즉, *C1*과 *C2*셀이 직접인접 관계에 있고 *C2*와 *C3*셀도 직접인접 관계에 있기 때문에, *C1*과 *C3*셀은 인접 가능하다.



(a) 대각선 인접 가능 (b) 수식 인접 가능 (c) 수직 인접 가능

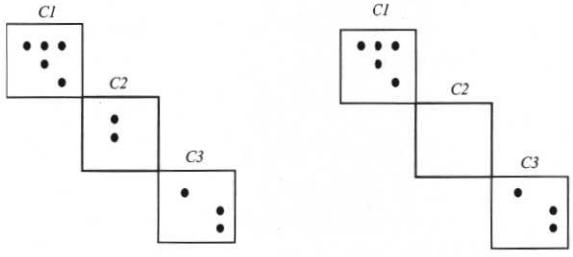
(그림 4) 인접 가능 예

인접 가능 정의를 기반으로 셀들간의 전파에 대하여 정의한다. 클러스터링 전파 여부는 인접 가능을 기반으로 한다. 전체 데이타 공간으로부터 의미 있는 부분집합을 발견하기 위한 필수 조건은 클러스터링 전파 가능 여부를 판단하는 것이다. 이것은 셀을 기준으로 전파를 시켜 전체 영역에 대하여 클러스터를 찾을 수 있다.

클러스터링 전파(Clustering Propagation) : 그리드 셀 구조에서 공간객체들이 존재하는 임의의 셀들 *C1*, *C2*, *C3*에 대하여 *C1*과 *C3*가 *C2*를 기반으로 하여 인접 가능하면, *C1*과 *C3* 셀간에는 클러스터링이 가능하다.

4.2 클러스터 생성 알고리즘

그리드 셀 기반 공간 클러스터링 알고리즘은 세부적으로 클러스터 생성 알고리즘과 클러스터 합병 알고리즘으로 구성된다. 클러스터링 생성 알고리즘은 셀 관련성을 이용하여 후보 클러스터들을 생성한다. 그리고 클러스터 합병 알고리즘은 전 단계에서 생성된 후보 클러스터들에 대하여 합병 가능 여부를 판단하여 합병한다.



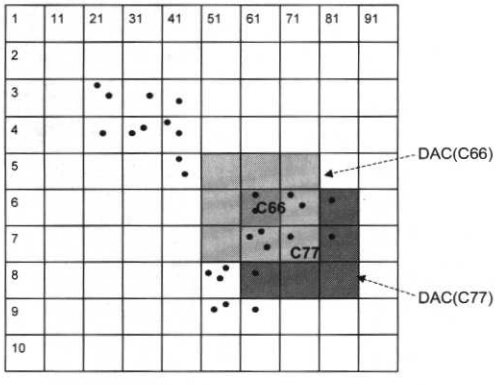
(a) 전파 가능 (b) 전파 불가능

(그림 5) 클러스터링 전파 예

클러스터 생성 알고리즘은 직접인접, 직접인접 셀, 인접 가능, 클러스터링 전파 정의를 이용하여 클러스터를 생성한다. 세부적으로 생성 알고리즘은 셀 관련성을 기반으로 전파 과정을 반복하여 클러스터를 생성한다. (그림 5)는 셀들간의 클러스터링 전파 가능 및 불가능한 예를 보여준다. 먼저 *C2*셀에 대해 *C1*과 *C3*셀은 셀 관련성에 의해 인접 가능하다. 그러나 실제 클러스터링은 셀 내의 객체들에 대하여 수행된다. 따라서 (그림 5)(a)에서 *C2*셀은 객체가 존재하므로 *C1*셀이 *C3*셀로 전파 가능하고 하나의 클러스터를 묶을 수 있다. 반면에 (그림 5)(b)에서는 *C2*셀에 객체가 존재하지 않으므로 *C1*셀과 *C3*셀의 객체들에 대해서는 하나의 클러스터로 구성할 수 없다.

생성 알고리즘의 클러스터링 과정은 먼저 그리드 셀 구조에서 객체가 존재하는 셀들에 대하여 셀 관련성에 의해 직접인접 셀들을 찾는다. 그리고 직접인접 셀들을 기반으로 클러스터링의 대상이 되는 셀들을 전파하여 후보클러스터를 찾는다. 이때 후보 대상이 되는 셀들은 인접 가능한 셀들이다. (그림 5)(a)에서 *C1*셀을 기준으로 하면 *C2*가 직접인접 셀이 되고, *C3*는 *C2*를 기반으로 하여 *C1*과 인접 가능 셀이 된다. 따라서 *C3*셀은 하나의 클러스터로 형성될 수 있는 후보 셀이 된다. 이때 중요한 것은 클러스터링할 때 기준은 셀 내에 포함되어 있는 객체들 간의 거리가 임계값 내에 있어야 된다는 것이다. 따라서 *C2*셀에 객체가 존재하면 셀 관련성에 의하여 *C3*셀의 객체들은 *C1*셀의 객체들과 하나의 클러스터를 구성한다. 반면에 *C2*셀에 객체가 존재하지 않으면, *C3*셀과 *C1*셀의 객체들간의 거리가 임계값 내에 있다는 것을 셀 관련성만으로는 보장할 수가 없다. 이러한 경우에는 직접 객체들간의 거리 계산을 통하여 클러스터링 여부를 판단할 수 밖에 없다.

(그림 6)은 전파를 이용한 클러스터링 생성을 보여준다. 여기서는 먼저 *C66*을 기준셀로 하여 직접인접 셀들인 *DAC*(*C66*)을 찾는다. 그리고 *DAC*(*C66*)의 셀들 중에서 객체들을 포함한 임의의 셀(*C77*)을 기준으로 하여 *DAC*(*C77*)을 찾는다. 더 이상 전파가 일어나지 않을 때까지 이 과정을 계속 반복하여 클러스터를 생성한다. (그림 6)의 클러스터링 결과는 (그림 8)과 같이 2개의 클러스터들이 생성된다.



(그림 6) 전파를 이용한 클러스터링

클러스터 생성 알고리즘은 (그림 7)과 같다. 이 알고리즘에서 먼저 *CreateDAC* 함수는 전체 셀들에 대하여 반복 수행한다. 여기서 임의의 셀에 대하여 객체가 존재하고 클러스터로 구성되지 않은 셀이면 *PropagateCluster* 함수를 호출한다. 이 함수에서 *CNearCell*은 현재 기준셀에 직접인

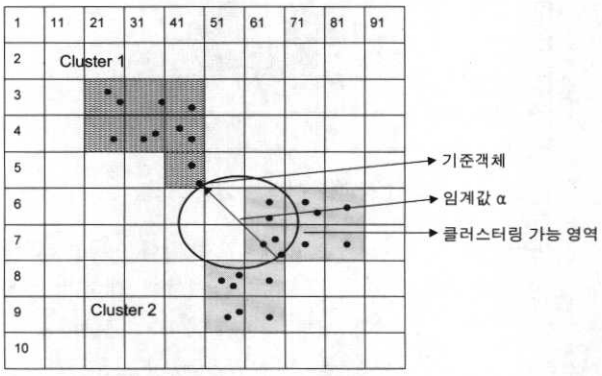
접한 셀들에 대한 정보를 가지는 클래스이다. *CNearCell*에 속하는 각 셀에 대하여 객체가 존재하고 클러스터로 구성되지 않으면 현재 클러스터 ID를 부여하여 클러스터에 포함시킨다. 그리고 이 셀에 대하여 *PropagateCluster* 함수의 재귀호출을 통하여 전파시켜 나가면서 클러스터링 과정을 반복적으로 수행한다.

```
void CreateDAC ( )
{ CELL_ID current_cell ;
    CLUSTER_ID CurrentMaxClusterID = 0 ;
  for (current_cell = 0 ; current_cell < cell_count ; current_cell++)
  { if (grid_directory [current_cell].GetCount ( ) == 0) continue ;
      if (grid_directory [current_cell].ClusterID != -1) continue ;
      grid_directory [current_cell].ClusterID = CurrentMaxClusterID ;
      PropagateCluster (current_cell, CurrentMaxClusterID) ;
      CurrentMaxClusterID ++ ; }
}
void PropagateCluster (CELL_ID current_cell,
                       CLUSTER_ID current_cluster_id)
{ CELL_ID near_cell ;
   CNearCell NearCell (current_cell) ;
   for (int i = 0 ; i < 8 ; i++)
   { if (NearCell.GetNear ( i, near_cell ))
      { if (grid_directory [near_cell].GetCount ( ) != 0)
          { if (cell_info [near_cell].cluster_id == -1)
        { grid_directory [near_cell].ClusterID = current_cluster_id ;
          PropagateCluster (near_cell, current_cluster_id) ; } } } }
}
```

(그림 7) 클러스터 생성 알고리즘

4.3 클러스터 합병 알고리즘

클러스터 생성 알고리즘에서 추출된 클러스터들은 단순히 셀 관련성을 이용하여 생성되었기 때문에 최종 결과는 아니다. 즉, 생성된 클러스터들에 포함된 객체간의 거리를 기준으로 보면 임계값 이내에 존재하는 객체들이 있을 수 있다. 예를 들어 (그림 8)에서 *Cluster 1*과 *Cluster 2*는 생성 알고리즘에서 별도의 클러스터들로 생성되었지만, *Cluster 1*에 속하는 한 임의의 객체를 기준으로 임계값 내의 영역을 설정하면 *Cluster 2*의 일부 객체들이 포함됨을 알 수 있다. 즉,



(그림 8) 클러스터링 가능 여부

객체간의 거리를 비교하면 *Cluster 1*과 *Cluster 2*는 하나의 클러스터로 구성되어야 한다. 클러스터 생성 알고리즘에서는 셀 관련성만을 이용하므로 이러한 점을 해결할 수 없다. 따라서 클러스터 생성 알고리즘에 의해 생성된 후보 클러스터들간의 합병 여부를 판단하고 클러스터를 합병하는 알고리즘이 필요하다.

본 논문에서는 이러한 후보 클러스터들의 합병 여부를 판단하고 합병을 수행하는 클러스터 합병 알고리즘을 제시한다. 먼저 합병 알고리즘을 적용하기 위한 클러스터링 가능 셀 범위와 클러스터링 가능셀에 대한 정의를 한다.

클러스터링 가능셀 범위(ECCA) : 그리드 셀 구조에서 임의의 한 셀에 대하여 다른 셀들 간의 최단거리가 임계값 이내에 있는 모든 셀들의 집합

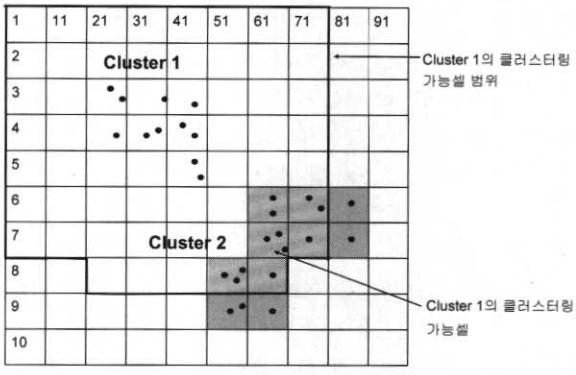
클러스터링 가능셀(ECC) : 클러스터의 클러스터링 가능 셀 범위의 셀들 중에서 다른 클러스터에 속해 있는 셀들의 집합

클러스터 합병 알고리즘에서 중요한 것은 전 단계에서 생성된 후보 클러스터들에 대하여 합병 여부를 판단하는 것이다. 이것은 셀 관련성으로는 해결이 안되므로 실제 클러스터들에 속하는 객체들간의 거리 계산을 통하여 판단해야 한다. 이 경우의 문제점은 후보 클러스터들간의 합병 여부를 판단하기 위하여 클러스터들에 속해 있는 객체들의 거리 계산에 대한 비용이 커진다는 것이다. 따라서 이 비용을 최소화시키는 것이 필요하다.

후보 클러스터들간의 합병 여부를 효율적으로 판단하기 위하여 먼저 거리 계산의 대상이 되는 셀들을 최소화해야 한다. 이를 위하여 임의의 셀을 기준으로 합병 가능한 셀들의 영역을 구한 후에, 이 영역에 속하는 셀들을 대상으로 거리 계산을 수행한다. 클러스터들의 합병을 위한 합병 가능 영역의 크기, 즉 클러스터링 가능 셀 범위는 클러스터의 가장자리를 기준으로 임계값 이내의 영역과 전부 또는 일부 포함되는 셀들의 집합이 된다. (그림 9)에서 *Cluster 1*의 경계를 기준으로 임계값을 이용한 클러스터링 가능셀 범위를 보여준다. 이 범위를 보면 *Cluster 2*의 6개의 셀들이 포함되어 있으며, 이 셀들이 *Cluster 1*의 클러스터링 가능셀이 된다.

클러스터 합병 알고리즘은 클러스터링 가능셀을 이용하여 합병 여부를 결정한다. 세부적으로 임의의 클러스터와 다른 클러스터와의 합병 여부는 클러스터에 속한 객체들과 클러스터링 가능셀에 포함된 객체들간의 거리를 비교하여 임계값 이내에 있으면 합병을 한다. 예를 들어, (그림 9)에서 *Cluster 1*과 *Cluster 2*의 합병 여부는 *Cluster 1*의 객체들과 클러스터링 가능셀에 포함된 객체들간의 거리를 비교한

다. 여기서 객체들간의 거리가 임계값 이내에 있는 객체들이 존재하므로 *Cluster 1*과 *Cluster 2*는 하나의 클러스터로 합병할 수 있다.



(그림 9) 클러스터링 가능셀 범위와 가능셀

클러스터 합병 알고리즘은 (그림 10)과 같다. 먼저 *Merge Cluster* 함수는 전체 셀들에 대하여 반복하여 수행하며, 후보 클러스터에 포함되는 셀이 검색되면 *SearchForNearCell* 함수를 호출한다. *SearchForNearCell*은 현재 주어진 셀의 주변 영역에 있는 임계값보다 작은 셀들을 대상으로 객체들 간의 거리 계산을 통하여 셀의 합병을 수행하는 함수이다. 이때 사용되는 *CECCANearCell*은 이 셀들에 대한 목록을 유지하기 위한 클래스이다. 또한, 이 알고리즘에서는 *PropagateECCA*를 이용하는데 이 함수는 현재 주어진 셀의 클러스터 ID와 동일한 셀을 찾아서, 이 셀들에 대하여 클러스터 ID를 전파하는 역할을 수행한다. 그러므로 합병 알고리즘은 주어진 클러스터에 대하여 실제 객체간의 비교 연산을 수행한 후에, 두 개의 클러스터가 합병 가능하여 동일한 클러스터로 인식되면 거리 연산은 더 이상 수행하지 않는다. 대신에 합병 가능한 셀들에 대하여 동일한 클러스터 ID를 부여하여 하나의 클러스터로 묶는다. 따라서 이 알고리즘은 객체간의 직접적인 연산은 가능한 최소화함으로써 알고리즘의 복잡성을 감소시킨다.

```
void MergeCluster ( )
{ CELL_ID current_cell ;
    CLUSTER_ID CurrentMaxClusterID = 0 ;
  for (current_cell = 0 ; current_cell < cell_count ; current_cell++)
  { if (grid_directory [current_cell].GetCount ( ) == 0) continue ;
    if (grid_directory [current_cell].ClusterID == -1) continue ;
    SearchNearCell(current_cell,grid_directory[current_cell].ClusterID) ; }
}
void SearchNearCell(CELL_ID CellID, CLUSTER_ID cluster_id)
{ CECCANearCell nearcells (CellID, x_size, y_size, x_count,
                           y_count, threshold) ;
  int num_near_cell = nearcells.getNearCellCount ( ) ;
  for (int i = 0 ; i < num_near_cell ; i++)
  { CELL_ID NearCellID ;
    nearcells.GetNearNthCell (i, NearCellID) ;
    if (grid_directory [NearCellID].GetCount ( ) == 0) continue ;
    if (grid_directory [NearCellID].ClusterID == cluster_id) continue ;
    CLUSTER_ID near_cluster_id
                = grid_directory [NearCellID].ClusterID ;
    double distance = GetMaxDistance (CellID, NearCellID) ;
    if (distance < threshold)
      { int ncluster_id = cell_info [NearCellID].cluster_id ;
        PropagateECCA(NearCellID,cluster_id) ; }
    distance = GetMinDistance (x, y, near_x, near_y) ;
    if (distance < threshold)
      { double obj_distance
                = GetObjMinDistance (CellID, NearCellID) ;
        if (obj_distance < threshold)
          PropagateECCA (ncluster_id, cluster_id) ; } }
}
```

(그림 10) 클러스터 합병 알고리즘

클러스터 합병 알고리즘을 수행하면 셀들로 구성된 클러스터가 생성된다. 마지막으로 클러스터를 구성하는 셀들 내에 있는 객체들을 추출하여 최종 클러스터들을 생성한다. 이것은 그리드 셀 구조를 생성할 때, 셀 내에 객체들의 정보를 가지고 있기 때문에 가능하다.

5. 실험 결과

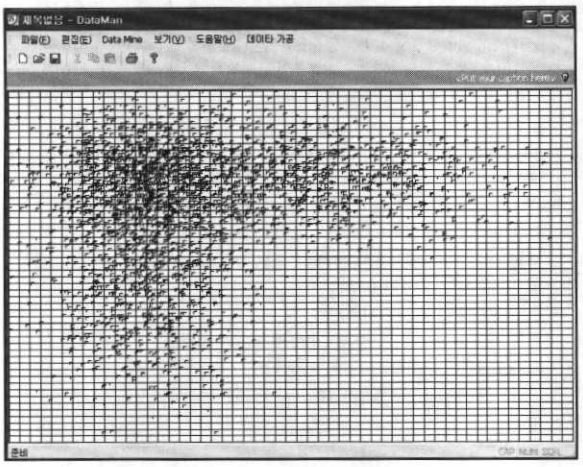
5.1 클러스터링 결과

실험 데이타는 각 축의 데이터 분포를 기준으로 생성하였으며, 실험의 공정성을 위하여 [14]의 Scholl Benchmark 데이타 집합으로부터 생성된 MBR 데이타의 좌표점을 객체의 위치로 하는 데이타를 생성하였다. 즉, 클러스터링을 위해 사용한 데이타는 점 객체를 대상으로 하였으며, 실험데이타에 대한 특성은 <표 1>과 같다. 여기서 DataSet 2는 DataSet 1과 데이터 분포 유형은 유사하지만, 데이터 객체 수는 차이가 난다. 그리고 실험 환경으로는 Pentium 4 2GHz, 메모리 512MB와 Windows XP 운영체제를 탑재한 PC를 기반으로 하였다.

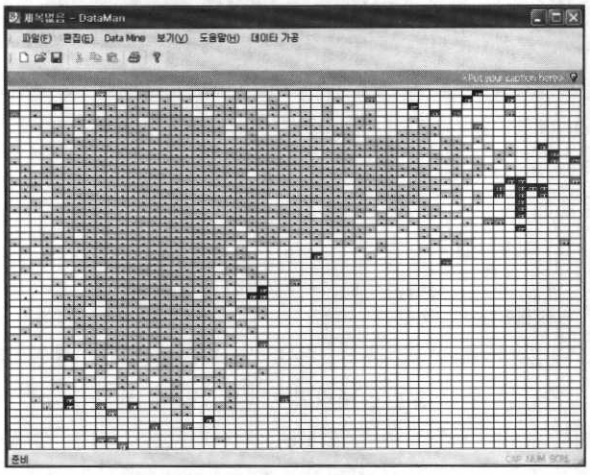
<표 1> 실험 데이타의 특성

데이타 집합	X축 범위	Y축 범위	X축 분포	Y축 분포	# of objects
DataSet1	0~10000	0~10000	GAUSSIAN	GAUSSIAN	12,000
DataSet2	0~10000	0~10000	GAUSSIAN	GAUSSIAN	16,000

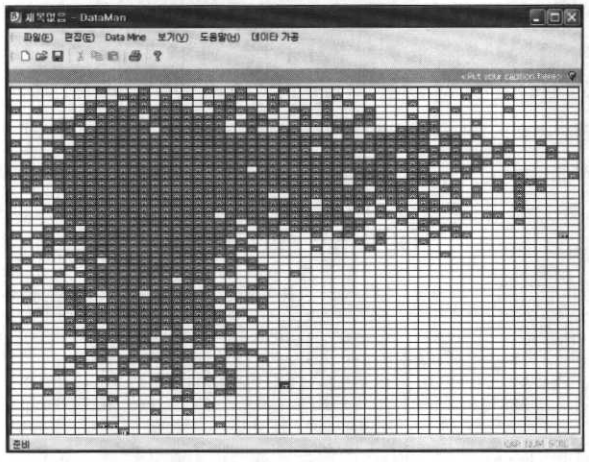
클러스터링 알고리즘을 수행하기 전에 먼저 그리드 셀 구조를 생성해야 한다. 이를 위하여 <표 1>의 DataSet 1 파일로부터 객체들을 읽어서 임계값을 기준으로 설정하여 그리드 셀 구조를 생성하고, 결과를 파일로 저장한다. (그림 11)은 DataSet 1에 대하여 임계값을 2000으로 설정하여 생성한 그리드 셀 구조와 객체들을 출력한 결과를 보여준다.



(그림 11) DataSet 1의 그리드 셀 구조 및 객체 출력



(그림 12) DataSet 1의 후보 클러스터들 생성 결과



(그림 13) DataSet 1의 후보 클러스터들 합병 결과

(그림 12)는 (그림 11)의 그리드 셀을 기반으로 생성 알고리즘을 적용하여 후보 클러스터들을 생성한 결과를 보여준다. 출력 결과에서 같은 색과 번호(Cluster ID)를 가지는 셀들을 동일한 클러스터를 나타낸다. 여기서 생성된 결과는 40개의 후보 클러스터들로서 군집을 형성한다. 좌측 상단의 경우에는 아주 밀집한 데이타 집합을 가져서 크기가 큰 하나의 클러스터로 형성된 것을 알 수 있었다. 반면에 주변 지역에는 데이터 밀집도가 떨어짐으로 인하여 크기가 작은 다수의 클러스터들 집합이 생성됨을 알 수 있다. (그림 13)은 (그림 12)의 후보 클러스터들에 대하여 합병 여부를 판단하여 합병한 클러스터들의 결과를 보여준다. 결과적으로 40개의 후보 클러스터들을 형성하던 DataSet 1은 4개의 클러스터들로 합병되었음을 알 수 있다.

5.2 성능 평가

알고리즘의 성능평가를 위하여 객체의 수에 따라 임계값을 변화시키면서 실험을 수행하였다. 공간객체의 수는 DataSet 1인 경우에는 12000개, DataSet 2인 경우에는 16000개를 대상으로 하였으며, 임계 값은 3500, 3000, 2500, 2000 값을 대상으로 하였다. 여기서는 임계 값이 작을수록 전체적인 셀들의 수는 많아지게 된다. DataSet 1, 2에 대한 전체적인 성능평가에 대한 결과는 <표 2>, <표 3>과 같다.

〈표 2〉 DataSet 1에 대한 성능 평가

(단위 : 초)

파일 이름	임계값	GridX	GridY	전체 셀	생성 시간	합병 시간	전체 시간
grid_30_30	3500	30	30	900	0.000192	0.070219	0.070411
grid_36_35	3000	36	35	1260	0.000278	0.127943	0.128221
grid_43_43	2500	43	43	1849	0.000377	0.24502	0.245397
grid_54_53	2000	54	53	2862	0.000549	0.516933	0.517482

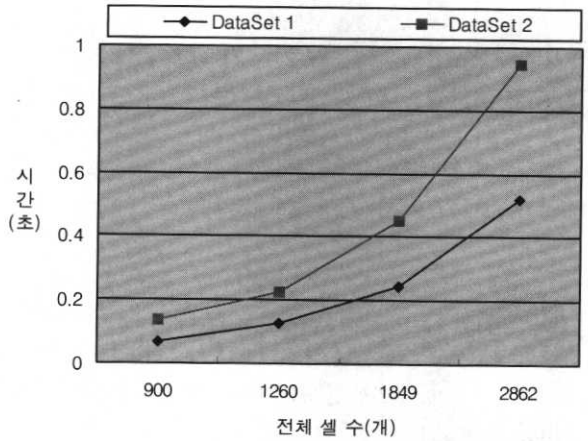
〈표 3〉 DataSet 2에 대한 성능 평가

(단위 : 초)

파일 이름	임계값	GridX	GridY	전체 셀	생성 시간	합병 시간	전체 시간
grid_30_30	3500	30	30	900	0.000284	0.135395	0.135679
grid_36_35	3000	36	35	1260	0.000368	0.22654	0.226908
grid_43_43	2500	43	43	1849	0.000504	0.45118	0.451684
grid_54_53	2000	54	53	2862	0.000753	0.946074	0.946827

DataSet 1과 DataSet 2를 대상으로 임계값 변화에 따라 증가된 전체 셀들을 대상으로 한 클러스터링 알고리즘의 수행 시간은 (그림 14)와 같다. DataSet 2가 DataSet 1에 비하여 시간이 증가하는 이유는 객체 수가 많기 때문이다. 실험 결과를 분석해 보면, 클러스터 생성 알고리즘과 합병 알고리즘은 전체 셀의 수에 따라 비례하여 증가하며, 또한 객체들의 수에 따라 비례하여 증가함을 알 수 있다. 여기서 주목할 것은 생성 알고리즘이 합병 알고리즘에 비하여 상대적으로 시간이 적게 들며, DataSet 1과 DataSet 2의 생성 시간을 비교하더라도 큰 차이가 없다. 이것은 생성 알고리즘이 셀들을 대상으로 셀 관계 연산만 수행하기 때문이다. 반면에 합병 알고리즘은 실제 객체들에 대한 연산을 수행하므로 DataSet 2가 DataSet 1에 비하여 시간이 증가한다.

또한 셀들 내에 있는 객체들을 대상으로 하므로 전체 셀의 수에 비례하여 시간이 증가한다.



(그림 14) 임계 값 변화에 따른 수행 시간 비교

성능평가를 검토해보면 전체 알고리즘 수행에서 클러스터 합병 알고리즘의 수행 시간이 차지하는 비중이 매우 크다. 이것은 상대적으로 클러스터 생성 알고리즘의 수행 시간이 적게 걸린다는 것을 의미한다. 즉, 전제적인 클러스터링 과정에서 셀 연산을 통하여 시간을 많이 줄였음을 알 수 있다. 그리고 클러스터 합병 알고리즘에서 실제 거리 계산의 대상이 되는 객체들은 클러스터링 가능셀에 포함된 것만을 대상으로 하기 때문에, 계산 시간이 많이 빨라졌음을 알 수 있다. 이것은 (그림 14)의 실험 결과에서 임계값에 따라 결과는 다소 차이가 있지만, DataSet 1은 12000개 객체들을 대상으로 전체적인 클러스터링 수행 시간이 약 0.52초 미만, DataSet 2는 16000개 객체들을 대상으로 약 0.95초 미만이 걸린 것을 통하여 확인할 수 있다.

6. 결론 및 향후 연구

공간데이터 마이닝을 위한 기존의 클러스터링 알고리즘들은 대부분이 객체들간의 거리 계산을 기반으로 하므로 데이타 양이 많아질수록 비용이 커진다. 또한 메모리 상주 데이타를 대상으로 하므로 대용량의 데이타인 경우에 효율이 떨어지는 문제점이 발생한다. 이러한 문제점들을 해결하기 위하여, 본 논문에서는 그리드 셀 구조를 기반으로 한 공간 클러스터링 알고리즘을 제시하였다. 세부적으로 셀 관련성을 기반으로 하여 후보 클러스터들을 생성하는 알고리즘과 후보 클러스터들의 합병 여부에 따라 클러스터를 합병하는 알고리즘을 제시하였다. 이 알고리즘에서는 기본적으로 셀 관련성을 기반으로 하여 실제 객체들간의 거리 계산을 최소화함으로써 대용량의 공간데이터에 대한 클러스터링에 효율적일 수 있다.

향후 연구로는 알고리즘의 성능을 입증하기 위하여 앞으로 다양한 분포 형태를 가진 데이터집합을 대상으로 실험을 계획 중이며, 특히 DBSCAN, CLARANS 등과 같은 기존의 클러스터링 기법들과의 성능 평가가 필요하다. 그리고 클러스터링의 기준이 되는 매개변수나 임계 값 도출에 있어서도 데이타 분포 상태, 밀도 분석을 위하여 반복적인 공간 분할을 이용하는 방법을 제시하고자 한다.

참 고 문 헌

[1] Ng and J. Han, "Efficient and Effective Clustering Method for Spatial Data Mining," Proc. of Int. Conf. on VLDB, pp. 144-155, 1994.

[2] M. Ester, H. P. Kriegel, J. Sander and X. Xu., "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise," Proc. of Int. Conf. on KDD, pp.226-231, 1996.

[3] W. Wang, J. Yang and R. Muntz, "STING : A Statistical Information Grid Approach to Spatial Data Mining," Proc. of Int'l Conf. on VLDB, pp.186-195, 1997.

[4] R. Agrawal, J. Gehrke, D. Gunopulos and P. Raghavan, "Automatic Subspace Clustering of High Dimensional Data Mining Application," Proc. of ACM SIGMOD, pp.94-105, 1998.

[5] Fayyad, U. M., et al., "Advances in Knowledge Discovery and Data Mining," AAAI Press/MIT Press, 1996.

[6] W. Lu, J. Han and B. C. Ooi, "Discovery of General Knowledge in Large Spatial Databases," Proc. of Far East Workshop on Geographic Information Systems, pp.275-289, 1993.

[7] Kaufman L. and Rousseeuw P. J., "Finding Groups in Data : an Introduction to Cluster Analysis," John Wiley & Sons, 1990.

[8] 오병우, 한기준, "H-SCAN : 지식 추출을 위한 해시-기반 공간 클러스터링 알고리즘", 정보과학회논문지, 제26권 제7호, pp.857-869, 1999.

[9] 진두석, 장재우 "데이타 마이닝을 위한 대용량 고차원 데이타의 셀-기반 분류방법", 정보과학회 학술발표논문집, 제27권 제2호, pp.192-194, 2000.

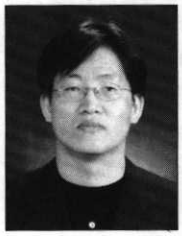
[10] 이혜명, 박영배, "고차원 데이타에서 점진적 프로젝션을 이용한 클러스터링", 정보과학회 학술발표논문집, 제27권 제2호, pp.189-191, 2000.

[11] 이동규, 정정수, 문상호, "셀-기반 공간클러스터링 방법", 정보과학회 학술발표논문집, 제28권 제1호, pp.10-12, 2001.

[12] 용환성, "데이터 마이닝", 그린출판사, 1998.

[13] 이동규, 문상호, "데이타베이스에서 클러스터 생성을 위한 그리드 셀-기반 알고리즘", 정보과학회 영남지부 학술발표논문집, 제9권 제1호, pp.153-158, 2001.

[14] Spatial Join Benchmarking home page (http://www.enst.fr/~bdtest/sigbench/index.html).



문 상 호

e-mail : shmoon87@pufs.ac.kr

1991년 부산대학교 컴퓨터공학과(공학사)

1994년 부산대학교 대학원 컴퓨터공학과 (공학석사)

1998년 부산대학교 대학원 컴퓨터공학과 (공학박사)

1998년~2002년 위덕대학교 컴퓨터멀티미디어공학부 조교수

2002년~현재 부산외국어대학교 컴퓨터공학부 조교수

관심분야 : 공간 DB, 공간뷰, 데이터마이닝, Mobile GIS, GIS 표준, 정보시스템 감리 등

이 동 규

e-mail : 1365blue@hanmail.net

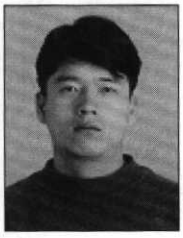
2000년 위덕대학교 컴퓨터공학과(공학사)

2002년 위덕대학교 일반대학원 정보전자공학과 컴퓨터공학전공(공학석사)

2000년~2001년 위덕대학교 컴퓨터공학과 조교

2002년~현재 AITSCOM(주) SI팀

관심분야 : 공간 DB, 데이터마이닝, EC/ERP 외



서 영 덕

e-mail : ydseo@pusan.ac.kr

1997년 부산대학교 컴퓨터공학과(공학사)

1999년 부산대학교 대학원 컴퓨터공학과 (공학석사)

1999년~현재 부산대학교 대학원 컴퓨터공학과 박사과정 재학중

관심분야 : 지리정보 시스템, 병렬 지리정보 시스템, 객체 지향 데이터베이스등