공간국부성을 최적화하는 클러스터링 방법
(A Clustering Method for Optimizing Spatial Locality)

김 홍 기 †
(Hong-ki Kim)

요 약 본 논문에서는 순환적인 검색공간과 장애물이 존재하는 검색공간에서 객체들을 클러스터링할 때 고려해야하는 CCD(Clustering with Circular Distance) 문제와 COD(Clustering with Obstructed Distance) 문제를 연구하였다. 그리고 다차원 검색공간에서 삽입이나 삭제가 빈번히 발생하는 객체들을 효율적으로 클러스터링하기 위한 새로운 클러스터링 알고리즘을 제안하였다. 제안한 클러스터링 알고리즘에는 CCD 및 COD 문제를 해결하기 위한 거리 함수가 정의된다. 그리고 최소의 연산 시간으로 높은 공간국부성을 갖는 클러스터들을 생성하기 위한 클러스터링 방법이 포함된다.

키워드 : 클러스터링, 순환거리, 장애거리, 공간국부성, 변형계층분산

Abstract In this paper, we study the CCD(Clustering with Circular Distance) and the COD(Clustering with Obstructed Distance) problems to be considered when objects are being clustered in a circularly search space and a search space with the presence of obstacles. We also propose a new clustering algorithm for clustering efficiently objects that the insertion or the deletion is occurring frequently in multi-dimensional search space. The distance function for solving the CCD and COD problems is defined in the proposed clustering algorithm. This algorithm is included a clustering method to create clusters that have a high spatial locality by minimum computation time.

Key words : Clustering, Circular Distance, Obstructed Distance, Spatial Locality, Modified Hierarchical Variance

1. 서 론

데이타마이닝, 지리정보시스템, 화상처리, 컴퓨터지원 설계 등과 같은 최근의 응용 분야에서는 더욱 복잡하고 많은 양의 공간적 성질을 갖는 객체를 취급하고 있다. 객체는 컴퓨터지원설계와 같이 삽입이나 삭제가 빈번히 발생하는 동적인 환경과 지리정보시스템과 같은 상대적으로 정적인 환경 모두에서 취급되어 진다. 따라서 이러한 시스템의 성능을 향상시키기 위해서는 동적 및 정적 환경에서 발생하는 객체를 효율적으로 관리하는 클러스터링 방법이 필요하다.

클러스터링 방법은 크게 분할(partitioning)[1-3], 계층(hierarchical)[4,5], 밀도 기반(density-based)[6,7] 그리고 그리드 기반(grid-based)[8] 방법 등으로 구분할 수 있다. 분할 방법은 클러스터 검출에서 가장 보편적으로 이용되는 방법으로 사전에 결정된 클러스터의 수 k에 기초하여 전체 객체를 상대적으로 유사한 k개의 클러스터로 구분하는 방법이다. 대표적인 분할 방법으로는 $k-means$ 방법[2]과 CLARA, CLARANS 등과 같은 $k-medoids$ 방법[1,3]이 있다.

분할 방법은 초기의 입력 벡터 개수에 따라 그룹화되는 객체들의 개수가 정해진다는 단점이 있다. 그러나 동적인 환경에서 특정 객체들을 유사한 객체들끼리 클러스터링 할 때에는 몇 개의 클러스터로 그룹화 해야 할지 사전에 정할 수 없는 경우가 대부분이다. 따라서 클러스터의 개수를 상황에 따라서 동적으로 결정할 수 있는 알고리즘이 필요하다.

다차원 검색공간에서 공간적으로 인접하는 객체들을 동일한 클러스터에 놓이게 함으로써 검색의 성능과 클러스터의 이용률을 향상시킬 수 있다. 효율적인 클러스터링이 이루어지기 위해서는 공간국부성(spatial locality) 정도가 반드시 고려되어야 하며, 이는 객체들 간의 거리와 관계가 있다.

기존의 클러스터링 방법은 객체의 속성이 가질 수 있는 최소와 최대값 사이의 선형적 순서 범위로 구성된 검색공간을 가정하였다. 그리고 알고리즘에서는 클러스

· 이 논문은 2003학년도 동신대학교 학술연구비에 의하여 연구되었음
† 종신회원 : 동신대학교 컴퓨터학과 교수
hkkim@dsu.ac.kr
논문접수 : 2003년 7월 21일
심사완료 : 2004년 1월 14일

터의 중심과 해당 클러스터에 포함된 객체들 간의 유클리드 거리(Euclidean distance)의 평균 또는 합을 최소화하는 전략을 갖는다.

그러나 대부분의 응용에서 가정하는 실세계의 검색공간은 다양할 수 있으며, 이러한 검색공간에서 유클리드 거리는 많은 결점을 가지고 있다. 예를 들면, 지리정보 시스템과 같은 응용에서 취급되는 객체 또는 시간의 개념이 포함되어 있는 객체 등은 순환적인 속성을 가질 수 있다. 클러스터링에서 이러한 객체의 속성을 효율적으로 반영하기 위해서는 순환적인 성질을 갖는 도메인으로 구성된 검색공간과 새로운 거리 측도가 요구된다.

또한 실세계의 검색공간 안에는 강, 호수 그리고 고속도로 등과 같은 여러 종류의 장애물이 존재하며, 이러한 환경에서 객체들을 클러스터링하기 위해서는 장애물을 고려한 객체 간의 거리 측도가 필요하다.

본 논문은 크게 두 가지 연구로 구분할 수 있다. 첫 번째는 순환적인 검색공간과 장애물이 존재하는 검색공간에서 객체들을 클러스터링할 때 고려해야하는 문제를 연구하는 것이다. 이 연구에서는 객체들 간의 거리 함수를 제안하고 관련 함수 값들을 최소화하는 방법을 이용하여 정의된 문제를 해결한다.

두 번째 연구는 효율적인 클러스터링 알고리즘을 설계하는 것이다. 알고리즘에서는 공간국부성 측도를 이용함으로써 계층구조를 갖는 클러스터들 간의 높은 공간국부성을 유지시킨다. 또한 최소의 연산 시간으로 클러스터를 구축 및 관리할 수 있는 방법을 연구한다.

이 연구는 공간 객체를 보조 기억장치에 저장하기 위한 저장시스템의 핵심이 되는 분야로 최근의 다양한 응용에서 발생하는 동적 및 정적 공간 객체를 효율적으로 관리하는데 기여한다.

논문의 구성은 다음과 같다. 2장에서는 다양한 검색공간에서 클러스터링에 고려해야 할 문제와 이를 해결하기위한 방법을 알아본다. 3장에서는 공간국부성 측도를 알아보고 2장에서 정의한 클러스터링 문제와 공간국부성 측도와의 관계에 대하여 논한다. 4장에서는 공간국부성을 고려한 새로운 클러스터링 알고리즘을 제안한다. 끝으로 5장에서 결론 및 앞으로의 연구 방향에 대하여 기술한다.

2. 검색공간과 클러스터링

클러스터링 방법은 임의의 측도에 의해 객체의 집합을 구분하여 서로 다른 그룹의 클러스터를 형성하는 것으로 여기에서 측도는 응용에 따라 다르다. 이 장에서는 실질적인 응용에서 고려할 수 있는 검색공간의 유형과 거리 측도를 알아보고, 다양한 검색공간에서 클러스터링에 고려해야 할 문제를 정의한다. 또한 정의된 문제를 해결하기 위한 기존의 클러스터링 알고리즘이 가지고 있는 문제점을 기술한다.

2.1 순환적인 검색공간

기존의 클러스터링 방법에서 N차원 검색공간을 구성하는 임의의 도메인 D_i ($i=1,2, \cdots, N$)은 해당 검색공간에 존재하는 객체들의 i번째 속성이 가질 수 있는 최소와 최대값 사이의 서로 다른 m개의 값들을 갖는 선형적 순서 범위로 이루어져 있다. 즉, $D_i = \{d_1, d_2, \cdots, d_m\}$이고, 여기에서 $d_j < d_{j+1}$, $j = 1,2, \cdots, m-1$이다.

그러나 객체들은 순환적인 위치 속성을 가질 수 있으며 클러스터링 방법에서 이러한 객체를 효율적으로 취급하기 위해서는 순환적 순서범위를 갖는 도메인으로 구성된 검색공간이 요구된다.

그림 1은 x축이 순환적 순서범위를 갖는 도메인으로 구성된 검색공간에서 객체 $P1$과 클러스터링될 대상 객체를 선택하는 예이다. $d(p,q)$를 객체 p와 q의 위치 속성에 따른 유클리드 거리라 하고, $d'(p,q)$를 순환적 순서범위를 갖는 도메인으로 구성된 검색공간에서 객체 p와 q 사이의 최소 유클리드 거리라 하자. 객체 $P1$과 클러스터링될 대상 객체로 $P2$ 또는 $P3$ 중 하나를 선택한다고 할 때, 만약 순환적 순서범위를 고려하지 않고 유클리드 거리를 측도로 사용하여 클러스터링한다면 유클리드 거리는 $d(P1,P2) < d(P1,P3)$이 됨으로써 객체 $P1$과 $P2$가 동일한 클러스터에 저장될 것이다. 그러나 순환적 순서범위를 고려한 최소 유클리드 거리는 $d'(P1,P2) > d'(P1,P3)$이 되기 때문에 객체 $P1$과 클러스터링될 대상 객체는 $P3$가 된다.

[9]에서는 순환적 순서범위를 갖는 순환 도메인(circular domain)을 다음과 같이 정의하였다.

객체의 위치 속성값을 매개 변수로 하여 임의 위치 속성값의 다음 또는 이전 위치 속성값을 반환하는 함수를 각각 *next*, *prev*라 하자. 도메인 D_i가 다음과 같은 성질을 만족할 때, D_i를 순환 도메인이라 한다.

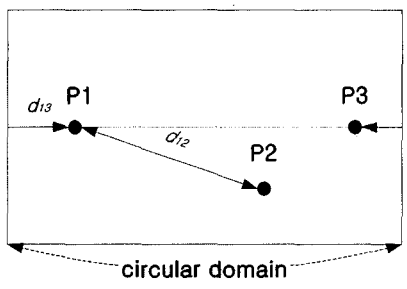

그림 1 순환적인 검색공간

- $next(d_k)=d_{k+1}$ $(1\le k< m)$ 그리고 $next(d_m)=d_1$
- $prev(d_k)=d_{k-1}$ $(1< k\le m)$ 그리고 $prev(d_1)=d_m$

본 논문에서는 순환 도메인으로 구성된 검색공간에서 클러스터링에 고려해야 할 문제와 이를 해결하기 위한 방법을 다음과 같이 제안한다.

정의 1. CCD(The Clustering with Circular Distance) 문제

다차원 검색공간 R에서 P를 n개 객체들의 위치 속성값에 따른 점 집합 $\{p_1, p_2, \cdots, p_n\}$이라 하자. 순환 도메인으로 구성된 R에서 객체 p와 q사이의 최소 유클리드 거리 $d(p,q)$를 순환 거리(circular distance)라 한다. CCD 문제는 P가 k개의 클러스터 $C_1, C_2, \cdots, C_k$로 클러스터링될 때 다음과 같은 에러 함수 E를 최소화하는 것이다.

$$E=\frac{1}{k}\sum_{i=1}^{k}\left\{\frac{1}{n(C_i)}\sum_{p\in C_i} d^2(p,g)\right\}$$

여기에서 $n(C_i)$: 클러스터 C_i에 포함된 객체의 수,
g : 클러스터 C_i의 중심.

2.2 장애물이 존재하는 검색공간

실세계의 검색공간 안에는 여러 종류의 장애물이 존재하며, 응용에서 이를 고려하지 않으면 올바르지 않은 클러스터링 결과로 인해 비효율적인 성능을 가져올 수 있다.

그림 2는 장애물이 존재하는 검색공간에서 객체 $P1$과 클러스터링될 대상 객체를 선택하는 예이다. $d'(p,q)$를 장애물에 방해받지 않는 객체 p와 q 사이의 최소 유클리드 거리라 하고 객체 $P1$과 클러스터링될 대상 객체로 $P2$ 또는 $P3$ 중 하나를 선택한다고 하자. 만약 장애물을 고려하지 않고 유클리드 거리를 측도로 사용하여 클러스터링한다면 유클리드 거리는 $d(P1,P2)< d(P1,P3)$이 됨으로써 객체 $P1$과 $P2$가 동일한 클러스터에 저장될 것이다. 그러나 장애물을 고려한 최소 유클리드 거리는 $d'(P1,P2)> d'(P1,P3)$이 되기 때문에 객체 $P1$과 클러스터링될 대상 객체는 $P3$가 된다.

이와 같이 검색공간에 장애물이 존재한다면 장애물의

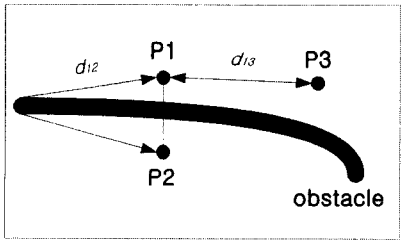

그림 2 장애물이 존재하는 검색공간

취급은 클러스터링 알고리즘에 있어서 중요한 요인이 된다. [10]에서는 장애물이 존재하는 환경에서 클러스터링에 고려해야 할 문제로 COD(Clustering with Obstructed Distance) 문제를 정의하였으며 이를 해결하기 위한 방법을 다음과 같이 제안하였다.

2차원 검색공간 R에서 P를 n개 객체들의 위치 속성값에 따른 점 집합 $\{p_1, p_2, \cdots, p_n\}$이라 하고 O를 겹치지 않는 m개의 장애물 집합 $\{o_1, o_2, \cdots, o_m\}$이라 하자. 객체 p와 q사이에서 O에 방해받지 않는 최소 유클리드 거리 $d(p,q)$를 장애 거리(obstructed distance)라 한다. COD 문제는 P를 k개의 클러스터 $C_1, C_2, \cdots, C_k$로 클러스터링할 때 다음과 같은 에러 함수 E를 최소화하는 것이다.

$$E=\sum_{i=1}^{k}\sum_{p\in C_i} d^2(p,m_i)$$

여기에서 m_i : 클러스터 C_i의 중심.

2.3 기존 클러스터링 알고리즘의 문제점

CCD 문제와 COD 문제는 기존의 클러스터링 알고리즘에 거리 함수를 변경하고 에러 함수 E를 최소화함으로써 해결할 수 있다. [11]에서는 CCD문제를 해결하기 위한 알고리즘을 R-트리에 적용한 동적 색인구조인 CR-트리(Circular R-Tree)를 제안하였다. CR-트리는 순환 도메인으로 구성된 다차원 검색공간에서 순환 거리를 이용하여 이웃하는 객체들을 효율적으로 클러스터링하기 위한 공간색인구조이다. CR-트리에서는 동적인 환경에서 발생하는 객체를 효율적으로 관리하기 위해 R-트리에 기초한 삽입, 삭제 알고리즘을 사용하여 에러 함수 E를 최소화하는 색인을 구축한다. 그러나 알고리즘에서는 클러스터링에 필요한 연산 시간을 최소화하지 못했다.

COD 문제를 해결하기 위해 효율적인 클러스터링 전략을 갖는 COD-CLARANS라는 $k-medoids$ 방법이 제안되었다. COD-CLARANS 알고리즘에서는 장애물을 취급하는 문제를 최적화시키기 위해 마이크로-클러스터라는 전처리 과정을 수행하면서 에러 함수 E를 재귀적으로 최소화 시켜나가는 기법을 적용하였다.

COD-CLARANS 알고리즘은 2차원 검색공간에서 객체의 삽입과 삭제가 자주 발생하지 않는 정적인 환경을 전제로 하고 있으며, 초기의 입력 벡터 개수에 따라 생성될 클러스터의 수가 정해진다. 그리고 COD 문제의 에러 함수 E에서는 각 클러스터에 저장되는 객체의 개수가 고려되지 않았다.

그러나 많은 응용에서는 동적인 환경을 전제로 하고 있으며, 객체들을 클러스터링할 때 몇 개의 클러스터로 그룹화해야 할지 사전에 결정할 수 없는 경우가 대부분

이다. 또한 클러스터링 결과 각 클러스터에 포함될 객체의 개수도 일정치 않을 수 있다. 따라서 다차원 검색공간에서 클러스터의 수를 상황에 따라서 동적으로 결정할 수 있는 효율적인 클러스터링 방법이 필요하다.

3. 공간국부성 측도

공간국부성은 객체들이 검색공간 상에 공간적으로 인접하는 정도를 나타내며 객체의 위치 속성에 따른 분산과 관계가 있다. 각 클러스터에 객체가 저장될 때 효율적인 클러스터링이 이루어지기 위해서는 공간국부성의 정도가 반드시 고려되어야 하며 이러한 공간국부성 정도를 측정하는 측도가 필요하다.

이 장에서는 [9]에서 제안한 공간국부성 측도인 전체분산(total variance)과 변형계층분산(modified hierarchical variance)을 소개한다. 그리고 2장에서 정의한 CCD 및 COD 문제와 전체분산과의 관계에 대하여 논한다.

전체분산은 검색공간상에 존재하는 객체들의 분포정도를 나타내고, 전체분산값이 낮을때 객체들 간의 공간국부성은 강하다. 전체분산 V_T^2은 다음과 같이 정의된다.

$$V_T^2 = \frac{1}{n}[P-G][P-G]^T$$

여기에서 $P=[p_1, p_2, \cdots, p_n]$($p_i$: i번째 객체의 공간적 위치),

$G=[g, g, \cdots, g]$(g: 검색공간에 존재하는 객체 집합의 중심).

객체가 저장될 클러스터의 크기를 고려하여 객체의 집합이 k개의 클러스터로 분할된다면 검색공간은 각 클러스터가 검색공간에 차지한 영역인 k개의 부검색공간으로 나누어지며, 이때 전체분산으로는 공간국부성 정도를 표현할 수 없게 된다. 이러한 문제는 k개의 클러스터에 따른 각각의 전체분산을 구하여 그 평균 또는 합을 이용함으로써 해결할 수 있다. 즉, k개의 클러스터에 따른 공간국부성 정도는 V_{Ti}^2를 i번째 클러스터의 전체분산이라고 할 때 $\frac{1}{k}\sum_{i=1}^{k} V_{Ti}^2$ (또는 $\sum_{i=1}^{k} V_{Ti}^2$)로 나타낼 수 있다. 따라서 높은 공간국부성을 갖는 클러스터를 유지하기 위해서는 $\frac{1}{k}\sum_{i=1}^{k} V_{Ti}^2$ 값을 최소화 시켜야 하는데 이는 CCD 문제의 에러 함수 E를 최소화하는 것과 동일하다.

그러나 COD 문제의 에러 함수 E에는 각 클러스터에 포함될 객체의 수가 고려되지 않았기 때문에 공간국부성 정도를 알 수 없다. 따라서 COD-CLARANS 알고리즘은 COD 문제의 에러 함수 E를 최소화하는 전략을 갖는다하더라도 최적의 클러스터링 결과를 얻을 수 없다.

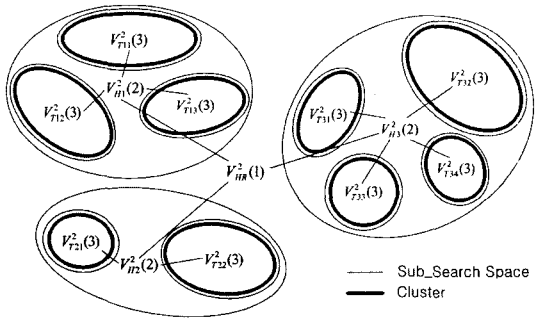

그림 3 검색공간과 변형계층분산

요약 정보 및 검색의 효율성 등을 위해 많은 응용에서는 검색공간을 하나 이상의 부검색공간을 갖는 계층구조로 표현한다. 변형계층분산은 임의 검색공간에 존재하는 모든 객체의 공간국부성과 부검색공간들 간의 평균 공간국부성을 고려한 계층별 공간국부성 측도이다. 그림 3에서는 객체가 저장되는 클러스터와 계층별 부검색공간에 따른 변형계층분산의 관계를 보인다. j계층의 임의 검색공간 R에 따른 변형계층분산 $V_{HR}^2(j)$은 다음과 같이 정의된다.

만약 R이 최하위 계층을 가리키는 검색공간이면

$$V_{HR}^2(j) = \frac{1}{k}\sum_{i=1}^{k} V_{Ti}^2(j+1) + V_{TR}^2(j)$$

그렇지 않으면

$$V_{HR}^2(j) = \frac{1}{k}\sum_{i=1}^{k} V_{Hi}^2(j+1) + V_{TR}^2(j)$$

여기에서 $V_{Ti}^2(j+1)$: i번째 클러스터의 전체분산,

$V_{Hi}^2(j+1)$: i번째 부검색공간의 변형계층분산,

$V_{TR}^2(j)$: 검색공간 R의 전체분산,

k : j계층의 클러스터 수 또는 부검색공간 수

COD-CLARANS에서는 객체들을 클러스터링함에 있어서 계층을 고려하지 않았다. CR-트리에서는 계층을 고려하였으나 클러스터링 결과 생성되는 부검색공간들의 겹침(overlap)을 허용하였다. 이는 최적의 공간국부성을 유지하는데 장애가 된다. 왜냐하면 부검색공간들이 서로 겹치게 된다면 겹치는 영역에 놓인 객체를 어느 클러스터에 속하게 하느냐에 따라 공간국부성의 정도가 달라지기 때문이다. 따라서 변형계층분산은 부검색공간의 겹침이 없을 때 정확한 계층별 공간국부성 측도가 된다.

4. 변형계층분산을 이용한 클러스터링

이 장에서는 삽입이나 삭제가 빈번히 발생하는 동적인 환경에서 객체들을 효율적으로 클러스터링하기 위한

새로운 클러스터링 알고리즘을 제안한다. 2.3절과 3장에 기술된 기존 클러스터링 알고리즘의 단점을 해결하기 위해 본 논문에서 고려한 클러스터링 알고리즘의 설계 원칙은 다음과 같다.

• 다양한 유형의 다차원 검색공간 고려

응용에서 가정하는 실세계의 다차원 검색공간은 다양할 수 있으며, 이러한 검색공간에서는 CCD 및 COD 문제와 같은 클러스터링에 고려해야할 다양한 문제가 정의될 수 있다.

• 계층을 고려한 높은 공간국부성을 갖는 클러스터 생성

검색공간은 겹침이 없는 하나 이상의 부검색공간을 갖는 계층 구조로 표현됨으로써 검색의 효율성을 높일 수 있으며 부수적으로 요약 정보도 얻을 수 있다. 그리고 이러한 계층 구조에서 높은 공간국부성을 갖는 클러스터들을 유지하기 위해서는 변형계층분산 값을 최소화하는 클러스터링 방법이 필요하다.

• 변형계층분산의 연산 시간 최소화

클러스터링 과정에서 최적의 공간국부성을 갖는 클러스터들을 생성하기 위해서는 변형계층분산에 따른 빈번한 연산이 요구된다. 변형계층분산은 전체분산들의 평균과 합으로 이루어져있다. 따라서 효율적인 알고리즘이 되기 위해서는 전체분산의 연산 시간을 최소화하는 방법이 필요하다.

4.1 거리 함수

CCD 및 COD 문제에서 순환 거리와 장애 거리 함수는 일반적인 유클리드 거리 함수를 변경함으로써 구현될 수 있다. 본 논문에서는 순환 도메인이 포함된 다차원 검색공간에서 객체 사이의 순환 거리를 다음과 같이 정의한다.

정의 2. $D_i=\{d_1, d_2, \cdots, d_m\}$(여기에서 $d_j<d_{j+1}$, $j=1,2,\cdots,m-1$)를 N차원 검색공간을 구성하는 i번째 도메인이라 할 때, N차원 검색공간에서 객체 p와 q 사이의 순환 거리 $d(p,q)$는 다음과 같다.

$$d(p,q)=\sqrt{\sum_{i=1}^{N} d_i^2(p,q)}$$

만약 D_i가 순환 도메인이면

$d_i(p,q) = \min(|d_p-d_q|, |d_p-d_1|+|d_q-d_m|)$

그렇지 않으면

$d_i(p,q) = |d_p-d_q|$

여기에서 $d_i(p,q)$: D_i에 사상된 객체 p와 q사이의 거리,

d_p, d_q: D_i에 사상된 객체 p, q의 속성 값 ($d_p<d_q$).

장애 거리는 가시 그래프(visibility graph)를 이용하여 구할 수 있다[10]. 가시 그래프 $VG=(V,E)$는 장애물들의 정점 집합 V와 서로 보이는 정점을 연결한 간선 집합 E로 구성된다. VG를 만들기 위해서는 장애물들의 정점을 조사하여 서로 보이는 모든 정점의 집합을 구해야 한다. 이는 두개의 객체가 서로 보이는가를 결정할 수 있는 데이타구조인 BSP-트리를 사용함으로써 해결할 수 있다.

객체 p와 q 사이의 장애 거리를 구하기 위해서는 VG의 확장된 가시 그래프인 $VG'=(V',E')$를 생성해야 한다. 가시 그래프 VG'에서 V'는 장애물들의 정점 집합 V에 객체 p, q의 위치를 포함시킨 것이고, E'는 V'에 있는 서로 보이는 정점들에 대한 간선 집합이다.

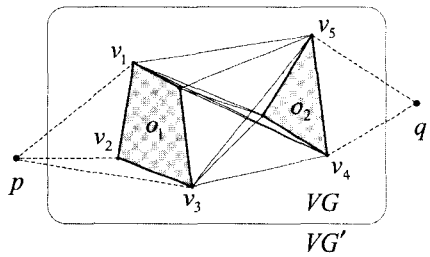

그림 4 가시 그래프

그림 4에서는 VG'가 두개의 장애물 o_1, o_2이 포함된 검색공간에 따른 VG에서 유도됨을 보인다. 객체 p에서 q까지의 장애 거리는 객체 p의 위치에서 시작해서 VG의 정점 v_1, v_2, v_3 중 하나를 경유하여 v_4 또는 v_5를 거쳐 객체 q의 위치까지의 최소의 값을 갖는 간선 경로이다.

검색공간에 다수의 고정된 장애물이 광범위하게 존재하는 경우에는 장애 거리를 구하기 전에 장애물에 따른 정점들의 전체 쌍을 대상으로 가시 그래프를 생성해 놓음으로써 중복 연산을 피할 수 있다. 그러나 장애물의 수가 적거나 고정되어 있지 않는다면 연산 시간을 줄이기 위해서 기준 반경 값 내에 놓이게 되는 정점들로만 대상을 제한하는 방법도 고려될 수 있을 것이다.

4.2 삽입 알고리즘

객체의 삽입에 따른 전략은 공간국부성 측면에서 매우 중요하다. 높은 공간국부성을 갖는 클러스터를 생성하고 유지하기 위해서는 변형계층분산 값을 최소화해야 한다. 변형계층분산은 전체분산들의 평균과 합으로 이루어져 있다.

따라서 객체의 삽입이 요구되면 삽입 알고리즘에서는 객체를 포함시켰을 때 전체분산의 증가가 최소가 되는 부검색공간을 경유하여 클러스터를 찾고, 해당 클러스터에 객체를 저장함으로써 높은 공간국부성을 갖는 클러스터를 생성할 수 있다.

만약 클러스터의 범람으로 부검색공간의 분할이 요구된다면, 부검색공간은 분할 후 생성될 두 개의 부검색공간에 따른 전체분산의 합이 최소가 되도록 분할된다.

```
Procedure INSERT(Object);
BEGIN
   Choose_Cluster(Object,Cluster);
   Insert_Object(Object,Cluster);
   IF Overflow(Cluster) THEN
      BEGIN
         Assign_New_Cluster(Cluster_N);
         Split_Cluster(Cluster,Cluster_N);
      END;
   Adjust_Tree();
END;
```

Choose_Cluster에서는 객체를 포함시켰을 때 전체분산의 증가가 최소가 되는 부검색공간을 찾기 위해 전체분산에 따른 연산이 빈번히 요구된다. 따라서 효율적인 삽입 알고리즘이 되기 위해서는 전체분산의 연산 시간을 줄일 수 있는 방법이 필요하다.

본 논문에서는 부검색공간에 기존의 정보(객체의 수, 객체집합의 중심, 전체분산)가 유지된다면, 객체의 삽입에 따른 부검색공간의 전체분산은 다음과 같이 유도된 수식 (1)을 이용함으로써 최소의 연산 시간으로 구할 수 있음을 보인다.

임의의 부검색공간에 따른 기존의 전체분산을 V_T^2라 할 때 새로운 객체 p가 해당 부검색공간에 삽입된다면 변경된 전체분산 $V_T^{2'}$은 다음과 같이 계산된다.

$$V_T^{2'} = \frac{1}{n+1}[P'-G'][P-G']^T$$

$$= \frac{1}{n+1}\sum_{i=1}^{n+1}((p_i-g)^2+2(p_i-g)(g-g')+(g-g')^2)$$

$$= \frac{n}{n+1}V_T^2+\frac{1}{n+1}(p-g)^2-\frac{2}{(n+1)^2}(p-g)^2+\frac{1}{(n+1)^2}(g-p)^2$$

$$= \frac{n}{n+1}V_T^2+\frac{n}{(n+1)^2}(p-g)^2 \quad (1)$$

여기에서 $P'=[p_1,p_2,\cdots,p_{n+1}]$, $p\in P'$

(p_i : i번째 객체의 공간적 위치),

$G'=[g',g',\cdots,g']$

(g' : 부검색공간에 존재하는 객체집합의 중심)

다음의 정리 1에서는 클러스터의 분할에 있어서 낮은 변형계층분산 값을 유지하기 위해서는 생성될 두 클러스터의 전체분산 합이 최소가 되어야 함을 보인다. 따라서 Split_Cluster에는 분할 후 생성될 두 클러스터의 전체분산 합이 최소가 되도록 하는 전략이 필요하다.

정리 1. 하나의 클러스터가 새로운 두 개의 클러스터로 분할될 때, 분할 후 생성된 두 개의 클러스터에 따른 전체분산의 합이 최소가 되면 변형계층분산의 증가는 최소가 된다.

증명. $j-1$레벨의 임의 검색공간 B가 t개의 부검색공간을 가지며 부검색공간 중 하나를 A라 하자. A가 최하위 레벨을 가리키는 부검색공간이라 할 때, A에 있는 k개의 클러스터 중 한 개의 클러스터가 분할된다면 분할 후 클러스터의 수는 $k+1$개가 된다. 클러스터의 분할 후에도 A의 전체분산 $V_{TA}^2(j)$에는 변화가 없다. 따라서 클러스터의 분할 후 변경된 A의 변형계층분산 $V_{HA}^{2'}(j)$은 다음과 같다.

$$V_{HA}^{2'}(j) = \frac{1}{k+1}\left\{\sum_{i=1}^{k-1}V_{Ti}^2(j+1)+\sum_{s=1}^{2}V_{Ts}^{2'}(j+1)\right\}+V_{TA}^2(j)$$

여기에서 $V_{Ts}^{2'}(j+1)$: 버켓 분할 후 생성된 s번째 버켓의 전체분산.

변형계층분산 $V_{HA}^{2'}(j)$이 최소가 되기 위해서는 분할 후 생성된 두 개의 $V_{Ts}^{2'}(j+1)$ 합이 최소가 되어야 한다. 또한 클러스터의 분할 후에도 A의 상위 레벨 검색공간 B의 전체분산에는 변화가 없음으로, 변경된 B의 변형계층분산 $V_{HB}^{2'}(j-1)$은 다음과 같이 표현된다.

$$V_{HB}^{2'}(j-1) = \frac{1}{t}\left\{\sum_{i=1}^{t-1}V_{Hi}^2(j)+V_{HA}^{2'}(j)\right\}+V_{TB}^2(j-1)$$

이는 해당 경로를 따라 최상위 레벨의 변형계층분산에 까지 계속 영향을 미친다. 따라서 분할 후 생성된 두 클러스터의 전체분산 합이 최소가 되면 변형계층분산의 증가는 최소가 된다. □

Split_Cluster에서는 범람이 발생한 클러스터에 존재하는 객체들 중에서 가장 멀리 떨어져 있는 두 개의 객체를 찾고 이를 분할 후 생성될 두 개의 클러스터에 할당한다. 그리고 할당되지 않은 각 객체에 대해 두 개의 클러스터에 각각 포함시켰을 때 전체분산의 증가가 최소가 되는 클러스터에 객체를 저장함으로써 변형계층분산의 증가를 최소화 시킬 수 있다.

4.3 삭제 알고리즘

객체의 삭제가 요구되면 삭제 알고리즘에서는 먼저 삭제하려는 객체가 포함된 클러스터를 찾고 해당 클러스터에서 객체를 삭제한다.

객체의 삭제로 인하여 클러스터에 있는 객체의 수가 최소 객체 수보다 작게 될 때, 알고리즘에서는 해당 클러스터에 이웃하는 부검색공간들을 조사하여 합병 후 최소의 전체분산 증분을 갖게 되는 부검색공간을 찾아 합병한다.

```
Procedure DELETE(Object);
BEGIN
   Find_Cluster(Object,Cluster);
   IF Cluster = null THEN exit;
```

```
Delete_Object(Object,Cluster);
IF Underflow(Cluster) THEN
BEGIN
  Find_Candidate_Cluster(Cluster,Cluster_C);
    Merge_Cluster(Cluster,Cluster_C);
  END;
Adjust_Tree();
END;
```

객체가 삭제되면 해당 클러스터의 상위 경로에 있는 부검색공간들의 기존 정보가 조정되어야 한다. 객체 삭제 후 부검색공간의 전체분산은 다음의 수식 (2)를 이용함으로써 최소의 연산 시간으로 갱신된다.

4.2절의 수식 (1)을 구하는 것과 동일한 방법으로, 부검색공간에 존재하는 객체 p가 삭제된다면 변경된 전체분산 $V_T^{2\prime}$은 다음과 같이 계산된다.

$$V_T^{2\prime} = \frac{n}{n-1} V_T^2 - \frac{n}{(n-1)^2}(p-g)^2 \quad (2)$$

Find_Candidate_Cluster에서 두 개의 부검색공간을 합병하여 하나의 부검색공간을 생성해 볼 때, 생성될 부검색공간의 전체분산을 구하기 위해서는 매우 많은 연산 시간이 요구된다. 왜냐하면 생성될 부검색공간에 포함될 모든 객체들을 조사해서 각 객체가 갖는 위치 속성값을 이용한 연산이 필요하기 때문이다. 다음의 수식 (3)은 부검색공간에 따른 기존의 정보를 이용함으로써 부검색공간의 합병시 전체분산의 연산 시간을 최소화할 수 있음을 보인다.

두 개의 부검색공간 A, B에 따른 전체분산을 각각 V_{TA}^2, V_{TB}^2라 하자. 그러면 V_{TA}^2, V_{TB}^2는 다음과 같이 기술할 수 있다.

$$V_{TA}^2 = \frac{1}{l}[P_A - G_A][P_A - G_A]^T$$

$$V_{TB}^2 = \frac{1}{m}[P_B - G_B][P_B - G_B]^T$$

여기에서,

$P_A = [p_{A_1}, p_{A_2}, \cdots, p_{A_l}]$, $P_B = [p_{B_1}, p_{B_2}, \cdots, p_{B_m}]$

(p_{A_i}, p_{B_i} : 각각 A, B에 포함된 i번째 객체의 공간적 위치),

$G_A = [g_A, g_A, \cdots, g_A]$, $G_B = [g_B, g_B, \cdots, g_B]$

(g_A, g_B : 각각 A, B에 존재하는 객체집합의 중심)

만약 C를 A, B를 합병 후 생성된 부검색공간이라 하면, C의 전체분산 V_{TC}^2은 다음과 같다.

$$V_{TC}^2 = \frac{1}{n}[P_C - G_C][P_C - G_C]^T$$

여기에서,

$P_C = [p_{C_1}, p_{C_2}, \cdots, p_{C_n}]$

(p_{C_i} : C에 포함된 i번째 객체의 공간적 위치),

$G_C = [g_C, g_C, \cdots, g_C]$

(g_C : C에 존재하는 객체집합의 중심).

그런데 C에 포함된 객체의 개수 n은 $l+m$과 같고, P_C는 $[p_{A_1}, p_{A_2}, \cdots, p_{A_l}, p_{B_1}, p_{B_2}, \cdots, p_{B_m}]$ 이기 때문에 V_{TC}^2는 다음과 같다.

$$\begin{aligned} V_{TC}^2 &= \frac{1}{l+m}[P_C - G_C][P_C - G_C]^T \\ &= \frac{1}{l+m}\sum_{i=1}^{l+m} p_{C_i}^2 - g_C^2 \\ &= \frac{1}{l+m}\left\{\sum_{i=1}^{l} p_{A_i}^2 + \sum_{i=1}^{m} p_{B_i}^2\right\} - \left\{\frac{1}{l+m}(l \cdot g_A + m \cdot g_B)\right\}^2 \\ &= \frac{1}{l+m}\{l(V_{TA}^2 + g_A^2) + m(V_{TB}^2 + g_B^2)\} \\ &\quad - \left\{\frac{1}{l+m}(l \cdot g_A + m \cdot g_B)\right\}^2 \end{aligned} \quad (3)$$

따라서 부검색공간 A, B가 합병되어 하나의 부검색공간 C를 생성한다면, 생성된 C의 전체분산 V_{TC}^2은 기존 A, B의 객체의 수 l, m과 객체집합의 중심 g_A, g_B 그리고 전체분산 V_{TA}^2, V_{TB}^2을 이용함으로써 최소의 연산 시간으로 구할 수 있다.

5. 결 론

분할 방법에 기초한 클러스터링 방법은 클러스터의 중심과 해당 클러스터에 포함된 객체들 간의 유클리드 거리의 평균 또는 합을 최소화하는 전략을 갖는다. 그러나 순환적인 검색공간과 장애물이 존재하는 검색공간에서 유클리드 거리는 많은 결점을 가지고 있다.

본 논문에서는 순환적인 검색공간에서 객체들을 클러스터링할 때 고려해야하는 CCD 문제를 정의하였다. CCD 문제는 클러스터링 알고리즘에 순환 거리 함수를 정의하고, CCD 에러 함수를 최소화함으로써 해결할 수 있다. CCD 문제의 에러 함수는 각 클러스터에 따른 전체분산의 평균과 같기 때문에 이를 최소화함으로써 높은 공간국부성을 갖는 클러스터들을 구축할 수 있다.

COD 문제는 장애물이 존재하는 검색공간에서 객체들을 클러스터링할 때 고려해야 하는 문제이다. CCD 문제와 마찬가지로 COD 문제는 장애 거리 함수를 정의하고 COD 에러 함수를 최소화함으로써 해결할 수 있다. COD 문제의 에러 함수에는 각 클러스터에 포함될 객체의 수가 고려되지 않았기 때문에 공간국부성 정도를 알 수 없다. 따라서 클러스터링 알고리즘이 COD 에러 함수를 최소화하는 전략을 갖는다하더라도 높은 공간국부성을 갖는 클러스터링 결과를 얻을 수 없다.

논문에서는 다차원 검색공간에서 객체들을 효율적으

로 클러스터링하기 위한 새로운 클러스터링 알고리즘을 제안하였으며 알고리즘 설계에 고려한 사항은 다음과 같다.

- 객체의 삽입이나 삭제가 빈번히 발생하는 동적인 환경
- CCD 및 COD 문제 해결을 위해 순환 거리와 장애 거리 함수 정의
- 변형계층분산을 최소화함으로써 계층 구조에서 클러스터들의 높은 공간국부성 유지
- 객체의 삽입, 삭제 그리고 클러스터의 분할 및 합병 따른 변형계층분산의 연산 시간 최소화

앞으로의 연구 과제는 논문에서 제안한 클러스터링 알고리즘을 구현하고 기존의 클러스터링 알고리즘과의 성능을 비교 평가하는 것이다.

참 고 문 헌

[1] L. Kaufman, P. J. Rousseeuw, Finding Groups in Data: An Introduction to Cluster Analysis, John Wiley & Sons, 1990.

[2] J. MacQueen, "Some Methods for Classification and Analysis of Multivariate Observations," Proc. of the 5th Berkeley Symp. Math. Statist, Prob., 1, pp.281-297, 1967.

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

[4] S. Guha, R. Rastogi, K. Shim, "CURE: An Efficient Clustering Algorithm for Large Databases," Proc. Int. Conf. on SIGMOD, pp.73-84, 1998.

[5] G. Karypis, E.-H. Han, V. Kumar, "CHAMELEON: A Hierarchical Clustering Algorithm Using Dynamic Modeling," COMPUTER, pp.68-75, 1999.

[6] M. Ankerst, M. Breunig, H.-P. Kriegel, J. Sander, "OPTICS: Ordering Points ToIdentify the Clustering Structure," Proc. Int. Conf. on SIGMOD, pp.49-60, 1999.

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

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

[9] 김홍기, 황부현, "순환도메인을 기반으로 하는 PR-화일의 구현 및 성능평가", 한국정보처리학회 논문지, 3권 1호, pp.63-76, 1996.

[10] A. K. H. Tung, J. Hou, J. Han, "Spatial Clustering in the Presence of Obstacles," Proc. Int. Conf. on Data Engineering, pp.359-367, 2001.

[11] 김홍기, 선휘준, "공간 데이타베이스 시스템에서 순환 속성을 지원하는 공간색인구조의 성능평가", 한국멀티미디어학회 논문지, 4권 3호, pp.197-204, 2001.

김 홍 기

1984년 전남대학교 계산통계학과(이학사). 1986년 전남대학교 대학원 계산통계학과(이학석사). 1996년 전남대학교 대학원 전산통계학과(이학박사). 1991년~현재 동신대학교 컴퓨터학과 부교수. 관심분야는 공간데이타베이스, 컴퓨터그래픽스, 멀티미디어시스템