고차원 데이터 패킹을 위한 주기적 편중 분할 방법

(A Cyclic Sliced Partitioning Method for Packing High-dimensional Data)

김 태완(Tae-Wan Kim)[1], 이 기준(Ki-Joune Li)[2]

요약.

이전의 연구들에서 제안된 많은 색인 방법들은 저차원과 동적인 환경을 가정하고 제안되었다. 그러나 최근의 많은 데이터베이스 응용분야들은 대용량, 고차원 그리고 정적인 환경에 대한 처리를 요구하고 있다. 따라서 기존의 저차원이고 동적인 환경에서 제안되었던 색인 구축 전략들은 특히 데이터 및 공간 분할에 있어서 새로운 환경에 잘 적응하지 못한다. 본 연구에서 우리는 이러한 사실들을 지적하였고, 새로운 환경에 적응하는 색인 구축 시 적용되는 새로운 분할 전략을 성능 모델에 근거하여 제안하였다. 우리의 접근 방법은 기본적으로 정적인 환경에서 색인 구축에 사용되는 패킹이라는 기법을 적용하였다. 그리고 고차원 환경에서 질의 성능의 기대값을 제시하는 민코프스키-합 비용모델에 대한 관찰 결과를 이용하였다. 이러한 것들에 바탕을 두어 우리는 데이터 및 공간을 균등하게 분할하는 것보다 불균등하게 분할하는 것이 좋을 것이라는 예측을 비용 모델에 대한 관찰 결과로써 도출하였다. 그리고 이러한 결과를 이용한 불균등 분할 방법과 성능 모델들을 제시하였다. 이 연구의 결론으로서 균등 분할 방법보다 불균등 분할 방법이 고차원 환경에서 더 효율적인 방법임을 성능 모델 및 실험을 통하여 보여주었다. 그리고, 어떻게 불균등하게 분할하는 것이 좋은지에 대한 명확한 계량적 기준들을 제시하였다.

국문 키워드:

고차원 데이터, 접근 방법, 패킹, 비용 모델

[1] 부산대학교 컴퓨터 및 정보통신연구소 전임연구원 이학박사
[2] 부산대학교 전자계산학과 부교수

Abstract.

Traditional works on indexing have been suggested for low dimensional data under dynamic environments. But recent database applications require efficient processing of huge size of high dimensional data under static environments. Thus many indexing strategies suggested especially in partitioning ones do not adapt to these new environments. In our study, we point out these facts and propose a new partitioning strategy, which complies with new applications' requirements and is derived from analysis. As a preliminary step to propose our method, we apply a packing technique on the one hand and exploit observations on the Minkowski-sum cost model on the other, under uniform data distribution. Observations predict that unbalanced partitioning strategy may be more query-efficient than balanced partitioning strategy for high dimensional data. Thus we propose our method, called CSP (Cyclic Spliced Partitioning method). Analysis on this method explicitly suggests metrics on how to partition high dimensional data. By the cost model, simulations, and experiments, we show excellent performance of our method over balanced strategy. By experimental studies on other indices and packing methods, we also show the superiority of our method.

Keywords:

High dimensional data, Access Method, Packing, Cost Model

1. 서론

최근 몇 년 동안 전통적인 데이터베이스 응용들과 여러 측면에서 기본적으로 다른 새로운 데이터베이스 응용들이 연구되어 왔다. 이러한 것들은 데이터웨어하우징, 멀티미디어 데이터베이스 시스템, 분자 생물학 등이다. 새로운 응용 분야들에서 데이터들의 공통적인 특징들은 대용량화된 데이터베이스, 다속성을 가지는 데이터, 그리고 읽기 위주의 환경들이다[4][11].

다속성을 가진 데이터는 일반적으로 특성 벡터라고 불리는 다차원의 점들로 특성추출 알고리즘 및 변환 방법들에 의하여 변환된다. 이때, 변환된 결과 벡터의 속성들의 개수를 일반적으로 차원이라 한다. 특성 벡터들에 대한 유사성 검색 또는 영역 질의는 이러한 응용 분야들에서 요구되는 기본적인 요구 사항들 중의 하나이며 특히 빠른 시간 내에 처리될 것을 요구한다[10].

많은 색인 방법들은 동적인 환경을 가정하고 특성 벡터들을 다루는 방법들을 제안하였다. 동적인 환경을 가정함으로써 이러한 색인 방법들은 (1) 과도한 데이터 삽입 시간의 초래 (2) 삽입 순서에 따른 색인 성능의 상이성 (3) 지역적인 분할 전략에 의한 비효율적인 노드 생성 (4) 비효율적인 노드 생성에 의한 불필요한 검색 비용의 초래 등과 같은 문제들을 대부분 내포하고 있다[13]. 이러한 문제들은 대부분 데이터에 대한 전처리를 통한 색인 구축방법에 의하여 해결될 수 있다. 이러한 전처리를 한 후, 색인을 구축하는 방법을 일반적으로 패킹이라 한다. 패킹은 과도한 삽입 시간 및 순서에 의한 민감성은 피할 수 있으나 효율적인 노드 생성을 위한 데이터 또는 공간 분할 전략들은 제안한 패킹 방법들마다 각각 다르다. 분할 전략들은 차원이 낮은 공간 데이터에 대한 색인 구축 방법들에서 제안된 영역 및 둘레의 합의 최소화 방법, 일차원적인 선형화를 이용한 방법, 데이터 분포를 고려한 방법 등이다. 균등 분포를 가정하는 경우 이들은 대부분 정방형성을 가지는 노드 생성에 목적을 두고 있다. 왜냐하면, 이차원에서 질의 창이 충분히 크지 않은 경우 정방형성을 가진 노드들이 영역 질의에 대하여 성능이 좋다고 많은 연구들에서 지적하였기 때문이다[5][19][24]. 이러한 정방형성은 고차원인 경우에는 영역 질의에 대하여 최악의 결과를 가져온다. 왜냐하면, 고차원인 경우 정방형성을 지향하는 분할 전략에 의하여 생성된 노드들의 한 변의 길이는 최선의 경우 전체 데이터 공간의 범위의 절반을 차지하는 1/2 이며 질의 창의 한 변의 길이는 차원이 높아질수록 1 에 가까워지기 때문이다. 보다 자세한 설명은 2 장을 참조하기 바란다. 이러한 이유 때문에 고차원 데이터를 위해서는 새로운 분할 전략이 제시되어야 한다.

본 논문의 구성은 다음과 같다. 2 장에서 이전에 제시된 패킹 방법들에 대한 소개 및 고차원 환경에 대하

여 간략하게 기술한다. 3 장에서는 질의 성능의 기대값을 예측하는 민코프스키-합 수식 모델을 소개하고 모델에 대한 관찰들을 제시한다. 그리고 이러한 관찰들에 의한 새로운 분할 방법을 제안하고 제안된 방법의 성능을 분석한다. 4 장에서 균등 분포를 가지는 합성 데이터에 대한 실험을 통하여 제시한 분할 방법의 우월성을 보여준다. 그리고 5 장에서 본 연구의 결론을 짓고 향후 연구에 대하여 기술한다.

2. 기존 연구 및 배경

이번 장에서는 본 연구에서 사용되는 용어들 및 가정들에 대하여 기술한 후, 기존의 패킹 방법들에 대한 간략한 소개 및 패킹과 관련된 고차원 공간의 특징들에 대하여 기술한다.

N개의 d-차원 점 데이터들이 $[0, 1]^d$ 하이퍼 단위 데이터 공간에 균등하게 분포되어 있다고 가정한다. 본 연구의 주제인 패킹은 주어진 데이터 집합을 전처리 과정을 거쳐서 P개의 노드들로 분할하여 효율적인 질의 처리를 제공하는 정적인 색인 구축 방법이다. 이때 노드는 보조 기억장치의 페이지와 일대일 대응 관계가 있기 때문에 노드에 삽입되는 데이터의 개수는 페이지 크기에 제한된다. 하나의 노드 또는 페이지에 삽입되는 최대 데이터의 개수를 블록킹 인수라 할 때, P 는 $\lceil N/$블록킹 인수$\rceil$ 이다. 노드는 d-차원의 최소 경계 하이퍼 입방체(MBC; Minimum Bounding hyper-Cube)로 표현하며 이는 각 변의 범위의 곱으로 표현된다. 따라서, P개의 MBC들의 집합 M 중 i 번째 MBC는 $[low_{i,1}, high_{i,1}] \times [low_{i,2}, high_{i,2}] \times \ldots \times [low_{i,d}, high_{i,d}]$ 로 표현된다. 영역 질의 $\vec{\mathbf{q}}$ 는 동일한 변의 길이 $L(q)$를 가진 d-차원의 하이퍼 입방체이며 MBC 표현과 유사하게 각 변의 범위의 곱으로 표현된다. 즉, $\vec{\mathbf{q}}$ 는 $[qlow_1, qhigh_1] \times [qlow_2, qhigh_2] \times \ldots \times [qlow_d, qhigh_d]$ 로 표현된다.

여태까지 제안된 패킹 방법들은 두 개의 주된 목적들이 있다. 하나는 빠른 색인 구축이고, 또 다른 하나는 빠른 질의 처리이다. 빠른 색인 구축을 위한 방법들은 주기억 장치 내에 할당된 버퍼들을 구조화하여 이용하는 버퍼-트리 방법들이 있다[2][3][9]. 동적으로 구축된 경우와 버퍼-트리를 이용하여 구축한 경우의 질의 성능은 거의 동일하다고 지적되었다[9]. 따라서 빠른 질의 처리 방법으로는 적당하지 않다. 이러한 방법들을 제외하고 대부분의 연구는 R-트리 계열의[16] 색인 구조를 이용한 정렬-분할 휴리스틱 방법들에 근거한 빠른 질의 처리 방법들에 집중되어 있다[8][14][15][19][22][26][27]. 이들은 크게 군집화를 이용한 방법[26], 공간 채움 곡선을 이용한 방법[19], 그리고 최소경계사각형을 이용한 방법들로[8][14][15][22][27] 구분될 수 있다.

2.1 기존 패킹 방법들

이번 절에서는 패킹 방법들 중에서 대표적인 방법들인 군집화를 이용한 방법, 공간채움곡선을 이용한 방법, 그리고 최소경계사각형을 이용한 방법들을 간단히 소개한다.

- 군집화를 이용한 방법(NNPack[26]): 데이터를 어느 한 축에 따라 정렬한 후, 블록킹 인수만큼의 최근접 이웃들을 찾는다. 찾아진 데이터를 이용하여 하나의 노드를 생성하고, 모든 노드들을 생성할 때까지 반복한다. 찾은 노드들을 이용하여 색인을 상향식으로 구축한다.
- 공간채움곡선을 이용한 방법(HP[19]): 데이터를 힐버트 순서화 값에 따라 정렬한 순서에 의하여 데이터를 블록킹 인수만큼 분할한다. 분할된 집합을 이용하여 노드를 구축한다. 색인은 구해진 노드들을 이용하여 상향식으로 구축한다.
- 최소경계사각형을 이용한 방법(STR[22]): N, d, 블록킹 인수, 그리고 P가 주어진다고 가정하자. 이때, STR은 우선 한 차원에 대하여 정렬한 후, 데이터를 한 차원에 대하여 $P^{1/d}$개로 분할한다. 데이터 집합을 $P^{1/d}$개로 분할하므로 분할된 집합이 포함하는 최대 데이터의 개수는 $\lceil N/P^{1/d} \rceil$이다. 차원 d를 $d-1$, 블록킹 인수$\times \lfloor P^{(d-1)/d} \rfloor$개의 데이터를 가지는 $P^{1/d}$개의 집합들에 대하여 위의 과정을 P개의 집합을 생성할 때까지 재귀적으로 반복한다. 분할된 P개의 집합들을 이용하여 노드들을 생성하고 생성된 노드들을 상위 레벨의 입력 데이터로 이용하여 색인을 상향식으로 구축한다.

군집화를 이용한 방법들 중에서 트리 형태의 색인에 군집화를 적용한 방법은 X-트리에서 사용되었다[7]. 이외에 군집화된 데이터의 양을 페이지의 크기에 한정 짓지 않은 많은 방법들이 제안되었다[18]. 공간채움곡선을 이용한 방법들 중에서 힐버트 순서화 이외에 Z-순서화 곡선을 이용한 패킹 방법 역시 제안되었다[12]. 최소경계사각형을 이용한 방법은 대개 동일한 데이터 개수 또는 동일한 영역 등으로 분할하는 방법들을 사용한다. S-트리와 VAM-Split R-트리 같은 경우[1][27], 불균등 분포를 가지는 데이터를 편중성 또는 분산값 등을 이용하여 데이터를 가능하면 균등하게 분할하려고 하는 방법들이다. VA-파일 같은 경우, 데이터를 균등 크기 또는 균등 분포를 가지는 그리드 형태로 분할한 방법이라 할 수 있다[28]. 다음 장들에서 소개되지만 이러한 균등성을 지향하는 분할 방법은 고차원 데이터를 위한 패킹 방법으로는 적당한 방법이 아니다. 왜냐하면, 차원이 높아질수록 데이터의 선택률이 낮다 하더라도 질의 창의 한 변의 크기가 길어지기 때문이다. 이러한 사실은 2.2 절에서 언급 하기로 한다.

R-트리 계열이 아닌 쿼드트리 및 그리드파일 등과 같은 색인들에 대한 패킹 방법들 역시 제안되었다

[17][21]. 색인을 구축한 후, 최적화 기법을 적용하여 노드를 효율적으로 재구성하는 패킹 방법들 역시 제안되었다[15][25]. 패킹 방법의 *NP-hardness*는 [25] 에서 증명되었다는 사실을 주목하기 바란다.

2.2 고차원 데이터 공간의 특징

데이터가 존재하는 공간의 차원이 높은 경우, 우리가 일반적으로 가지고 있는 직관들을 벗어나는 현상들이 존재한다. 예들 들면, 대부분의 데이터들은 데이터 공간의 표면부근에 위치하며 데이터의 분포는 아주 성기다는 사실, 어떤 한 점으로부터 최근접 거리와 최원접 거리의 차이가 없어진다는 사실, 그리고 차원에 대하여 지수적으로 문제의 복잡도가 증가한다는 사실 등등이다[10]. 이러한 현상들을 문헌들에서는 "차원의 저주(Curse of Dimensionality)"라고 언급하며 Bellman의 책에서 처음으로 소개 되었다[6].

차원이 낮은 경우 일반적으로 통용되는 사실들이 차원이 높아짐으로써 달라지는 사실들 중 본 연구의 패킹과 관련 있는 분할 개수 및 기하적 확률에 대하여 정리하면 다음과 같다.

- **분할 개수:** 데이터를 분할하는 가장 간단한 방법은 데이터를 각 차원에 대하여 균등하게 분할하는 방법이다. 차원 d가 주어진 경우, 각 차원에 대하여 한 번씩 균등하게 분할하는 경우 전체 2^d개의 노드들을 생성하게 된다. 예를 들어, 차원이 20차원인 경우 이러한 방법에 의하여 생성하는 노드의 개수는 2^{20}개이며, 이 때 필요한 데이터의 개수는 최대블록킹 인수·2^{20}이다. 데이터의 개수가 충분하지 않은 경우 몇몇 차원에서 분할을 포기하거나 또는 차원이 증가할 때 데이터의 개수가 지수적으로 증가해야 한다. 전자의 경우가 보다 현실적인 경우이며 차원이 20차원 이상인 경우 가장 단순한 이진 분할을 적용하더라도 현실적으로 모든 차원에 대한 분할을 어렵게 만든다. 만약 모든 차원들을 분할 축으로 선택하여 분할을 수행하는 경우 생성된 MBC들의 각 변의 길이는 데이터 공간의 범위의 절반인 1/2 이 된다. 어떤 차원을 분할 차원으로 선택해야 질의에 효율적인가에 대한 연구는 아직 깊이 있게 이루어지지 않았다는 사실을 주목하기 바란다.
- **기하적 확률:** 데이터가 d-차원 단위 입방체 $[0,1]^d$ 에 균등 분포되어 있다고 가정한 경우, 질의 창의 체적이 전체 데이터 공간의 체적에서 차지하는 비율을 기하적 확률 또는 선택률이라 한다. 이러한 값의 의미는 전체 데이터 개수에서 질의 창과 교차하는 데이터 개수의 비율을 의미한다. 따라서, 단위 데이터 공간에서의 기하적 확률은 질의 창의 체적인 $L(q)^d$가 된다. 선택률이 주어진다고 가정한

경우 차원이 증가하면 $L(q)$의 길이는 아주 빨리 데이터 공간의 전체 범위인 1에 가까운 값을 가진다. 예를 들면, 선택률이 0.1% 이고 차원이 10차원인 경우, $L(q)$의 길이는 0.501이다. 만약 차원이 20차원인 경우, $L(q)$는 0.708이다. 이때, 생성된 MBC들의 변의 길이가 1/2이면 이러한 질의 창은 생성된 모든 MBC들과 교차한다.

위와 같은 이유들 때문에 고차원인 경우 정방형이 아닌 MBC를 생성하는 분할 방법을 제안하여야만 한다. 이는 차원이 높은 경우 낮은 선택률이라 하더라도 질의 변의 길이가 0.5보다 큰 경우가 많기 때문이다.

3. 불균등 분할 방법

3.1 민코프스키-합 비용 모델에 대한 관찰

분할에서 가장 중요한 결정들은 분할 차원의 선택과 선택된 차원에서 분할 개수의 선택이라고 알려졌다. 차원이 낮은 경우에는 모든 차원에 대하여 균등성을 제공하는 차원 및 개수의 선택이 질의 성능에 효율적이라고 알려져 왔다[5][19][24]. 따라서, 균등한 크기의 그리드같은 타일링 또는 균등 깊이(*equi-depth*) 분할 방법이 차원이 낮은 경우 좋은 방법이다. 그러나, 낮은 차원에서 좋은 성능을 보여주는 위와 같은 방법은 고차원인 경우 더 이상 좋은 선택 방법이 아니라는 사실을 이전 장에서 지적하였다. 이러한 현상은 기하적 확률을 일반화한 민코우스키-합 비용 모델에 의하여 보다 자세하게 설명된다[8][23].

정의 1 (민코프스키-합 비용 모델). 한 변의 길이가 $L(q)$인 입방체 질의 $\bar{\mathbf{q}}$와 MBC들의 집합 M이 주어진 경우, 집합 M에 대하여 질의 창 $\bar{\mathbf{q}}$가 교차할 MBC들의 개수의 기대값 $E(M, \bar{\mathbf{q}})$는 아래와 같다.

$$E(M, \bar{q}) = \sum_{i=1}^{m} \prod_{j=0}^{d-1} \frac{\min(high_{i,j}, 1 - L(q)) - \max(low_{i,j} - L(q), 0)}{1 - L(q)} \quad (1)$$

M : MBC들의 집합이며, $|M| = P$이다.

$high_{i,j}$: i번째 MBC의 j번째 차원의 범위의 최대값.

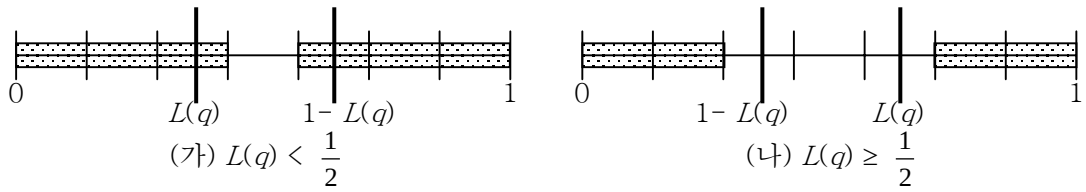
$low_{i,j}$: i번째 MBC의 j번째 차원의 범위의 최소값.

질의 입방체의 한 변의 길이 $L(q)$가 0.5인 경우, 균등분할 방법에 의하여 생성된 MBC들과 질의 창이 교차할 기대값을 위의 수식 (1)에 의하여 구하면 생성된 전체 MBC들의 개수인 P 이다. 따라서, $L(q)$의 값이 0.5

보다 큰 경우의 질의는 균등분할 방법에 의하여 생성된 모든 MBC들과 교차한다.

이제 수식 (1)에 대하여 자세한 관찰을 해 보자. 수식 (1)에서 집합 M에 대하여 질의 $\vec{\mathbf{q}}$가 교차할 MBC들의 개수의 기대값은 질의 $\vec{\mathbf{q}}$가 각 MBC와 교차할 기대값의 합으로 표현된다. 각 MBC의 기대값은 MBC의 각 차원에서의 변과 질의 $\vec{\mathbf{q}}$의 각 변에 의하여 구해진 기대값들의 곱으로 표현된다. 이러한 변들의 기대값은 MBC의 한 차원의 변의 위치(즉, $high_{i,j}$ 및 $low_{i,j}$의 위치)와 질의 변의 길이(즉, $L(q)$)에 의하여 결정된 값들의 곱으로 표현됨을 알 수 있다. 이때, MBC의 각 변의 기대값은 질의 길이에 의하여 분할된 데이터 공간에서 위치하는 곳에 따라 다르다. 따라서, 집합 M에 대한 $\vec{\mathbf{q}}$의 기대값은 각 MBC의 각 변의 기대값을 구하고 구해진 변들의 기대값들의 곱을 이용하여 한 MBC의 기대값을 구한다. 그리고 구해진 각 MBC의 기대값의 합이 집합 M에 대하여 $\vec{\mathbf{q}}$가 교차할 MBC들의 개수에 대한 기대값이다. 이러한 사실을 보다 명확하게 보여주기 위하여 다음의 예 3.1을 보자.

예제 3.1. 설명의 단순화를 위하여 MBC가 일차원인 경우에 대하여 설명한다. (그림 1)은 길이가 $1/P(=\delta)$인 $P(=7)$개의 MBC들 및 질의 변의 길이 $L(q)$에 의하여 분할된 데이터 공간을 보여준다. $L(q)$는 데이터 공간을 세 부분으로 분할한다. 이 때 점이 있는 블록은 $L(q)$가 0.5보다 작은 경우 $[0, L(q)]$ 및 $[1-L(q), 1]$와 교차하는 분할 공간을 나타내며 $L(q)$가 0.5보다 크거나 같은 경우 $[0, 1-L(q)]$ 및 $[L(q), 1]$에 포함된 분할 공간을 나타낸다.



(그림 1) $L(q)$에 의하여 분할된 공간 및 7개의 변들

$L(q)$가 0.5보다 작은 경우에는 (그림 1)의 (가)처럼 $[0, L(q)]$, $[L(q), 1-L(q)]$, 그리고, $[1-L(q), 1]$들로 분할한다. 이때, $[0, L(q)]$, $[L(q), 1-L(q)]$ 그리고 $[1-L(q), 1]$와 교차하는 MBC들의 기대값은 각각 다르다. 범위 $[0, L(q)]$와 **교차**하는 세 개의 MBC들의 기대값은 수식 (1)에 의하여 각각 δ / $(1-L(q))$, $2\cdot\delta$ /$(1-L(q))$ 그리고 $3\cdot\delta$ /$(1-L(q))$이다. 범위 $[1-L(q), 1]$와 교차하는 세 개의 MBC들의 기대값 또한 δ / $(1-L(q))$, $2\cdot\delta$ / $(1-L(q))$ 그리고

$3 \cdot \delta$ / (1–$L(q)$)이다. 범위 [$L(q)$, 1–$L(q)$]에 **포함**되는 MBC의 기대값은 모두 동일하며 (δ + $L(q)$)/(1–$L(q)$)이다. 질의 변의 길이 $L(q)$가 0.5보다 큰 경우 (그림 1)의 (나)와 같은 분할된 범위들을 가진다. 0.5보다 작은 경우와 비슷하게 MBC들이 [0, $L(q)$] 및 [1–$L(q)$, 1]에 **포함**되는 경우와 [$L(q)$, 1–$L(q)$]와 **교차**하는 경우에 따라 MBC들의 기대값은 각각 다르다. [0, $L(q)$] 및 [1–$L(q)$, 1]에 포함되는 MBC들의 기대값은 동일하며 이들의 합은 $2(1+2) \cdot \delta$ / (1–$L(q)$) 이다. $L(q)$가 0.5보다 큰 경우, [$L(q)$, 1–$L(q)$]와 교차하는 MBC들은 $L(q)$가 0.5보다 작은 경우와는 다르게 기대값이 모두 1이다. $L(q)$가 0.3 및 0.7인 경우 기대값은 각각 3.08개와 5.86개 이다. 이 값은 질의와 교차하는 MBC들의 기대 개수를 의미한다. □

예제 3.1에 의하여 우리는 두 가지 사실을 알 수 있다. (1) 질의 변 $L(q)$는 데이터 공간을 세 부분으로 분할하며 그 중 가운데 위치한 공간과 교차하는 MBC들의 변들의 기대값은 동일하다. 이때, 각 변은 최대 기대값을 가진다. (2) 데이터 공간의 양쪽 경계들에 위치한 MBC들의 변들의 기대값은 중앙에 위치한 변들의 기대값보다 작다. 그리고, 경계들에 가장 가까이 위치한 변의 기대값이 최소값을 가진다. 따라서, 분할을 하는 경우 가능하면 중앙 부분보다는 경계의 양 끝 부분에 더 많이 분할하는 것이 질의에 대하여 작은 기대값을 얻을 수 있다. 질의에 대한 일차원 MBC들의 기대값은 질의 변에 의하여 분할된 공간들과 MBC의 변들의 위치에 의하여 각각 다르게 됨을 위의 예를 통하여 알 수 있다. 이러한 사실을 보다 일반화 하여 $L(q) \geq 0.5$ 인 경우 d 차원의 MBC의 기대값에 대하여 알아 보기로 하자.

명제 3.1 d 차원의 MBC들과 $L(q) \geq 0.5$인 질의 입방체 $\vec{\mathbf{q}}$ 가 주어진다고 가정하자. i번째 MBC의 j번째 차원의 변 $I_{i,j}$를 [$low_{i,j}$, $high_{i,j}$]라 하자. 그러면, 이 변의 기대값은 아래와 같다.

(1) $high_{i,j} < 1 - L(q)$ 인 경우, $I_{i,j}$의 기대값은 $high_{i,j}/(1 - L(q))$.

(2) $low_{i,j} > L(q)$ 인 경우, $I_{i,j}$의 기대값은 $(1 - low_{i,j})/(1 - L(q))$.

(3) $high_{i,j} \geq 1 - L(q)$ 또는 $low_{i,j} \leq L(q)$ 인 경우, $I_{i,j}$의 기대값은 1 이다.

(4) $I_{i,j}$의 길이가 $1 - L(q)$보다 크거나 같은 경우, $I_{i,j}$의 기대값은 1 이다.

증명. 위의 네 가지 명제들과 관련된 증명은 [20]에서 보여주고 있다. □

Basic_Partition(D, s, *fanout*)
Input: D : a data set, s : a split dimension, *fanout* : the blocking factor
Output: D, MBC

1. let N be $|D|$ and D_1 be empty;
2. sort D by s in ascending order;
3. let s^{th} dimensional $(fanout - 1)^{th}$ value be $high_{s,left}$;
4. let s^{th} dimensional $(N - fanout)^{th}$ value be $low_{s,right}$;
5. if ($high_{s,left} < 1 - low_{s,right}$)
6. retrieve from 0^{th} to $(fanout - 1)^{th}$ data and insert into D_1 ;
7. else
8. retrieve from $(N - fanout)^{th}$ to $(N - 1)^{th}$ data and insert into D_1 ;
9. construct a MBC using D_1;
10. let D be $D - D_1$;
11. return D and MBC;

(그림 2) CSP의 기본 분할 알고리즘

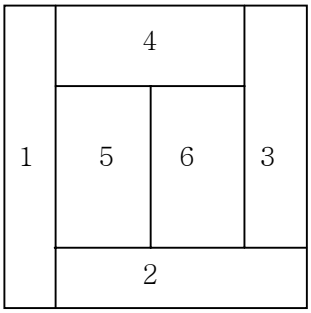
위의 명제 3.1에서 MBC 변들의 양 끝 값의 위치가 데이터 공간의 경계에 위치한 경우, 중앙 부근에 위치한 것보다 기대값이 작다는 사실을 보다 명확하게 보여준다. 이러한 사실을 분할 전략에 적용하면, 균등 분포된 데이터를 균등하게 분할하는 경우 MBC들의 변의 대부분은 데이터 공간의 중앙 부근에 위치하는 결과를 초래할 것이다. 따라서, 변들은 대부분 최대의 기대값을 가진다. 반면에 데이터 공간의 양 경계들에 위치한 데이터부터 분할하는 불균등한 분할 전략을 적용하면 작은 기대값을 가지는 범위들로 구성된 MBC들을 생성할 수 있을 것이다. 따라서 균등 분할보다 작은 기대값을 가지는 분할 방법이 될 것이다. 다음 절들에서는 이러한 사실에 근거한 불균등 분할 방법인 CSP 방법을 제안하고, 그 성능을 분석한다.

3.2 CSP 분할 방법

이번 절에서 우리는 각 차원에 대하여 주기적으로 일정한 부피만큼 데이터 공간을 분할하는 불균등 분할 방법인 *CSP*(Cyclic Sliced Partition) 방법을 제안하고 제안한 방법의 성능에 대하여 분석한다.

3.2.1 CSP 분할 방법

차원 i ($1 \le i \le d$) 에서 데이터 집합 D로부터 값이 작은 순서 및 큰 순서로 블록킹 인수 만큼의 데이터를 각각 추출한다. 작은 순서로 추출된 데이터를 감싸는 *MBC*의 i번째 차원의 범위를 $[low_{i,left}, high_{i,left}]$라 하고 큰 순서로 추출된 데이터를 감싸는 *MBC*의 i번째 차원의 범위를 $[low_{i,right}, high_{i,right}]$라 하자. 이 때, 두 개의 거리값 $high_{i,left}$ 와 $1.0 - low_{i,right}$ 들을 기억한다. 구해진 거리값들 중 최소값을 가지는 차원 및 방향에 따라 집합 D를 두 개의 부집합 D_1 과 D_2로 분할한다. 이때, 블록킹 인수 만큼의 데이터를 가지는 집합 D_1에 속한 데이터는



(그림 3) CSP 분할 순서 및 모양

D_2에 속한 모든 데이터 보다 i차원에서의 값이 모두 작거나 또는 모두 크다. 집합 D_1을 감싸는 노드를 생성하고 D_2를 D라 하고 차원 i를 $(i+1)\%d + 1$ 차원이라 하고 P개의 노드를 생성할 때까지 재귀적으로 CSP 분할 방법을 수행한다. CSP 분할 방법의 기본적인 분할 방법을 (그림 2)에서 보다 공식적으로 보여주었다. CSP 분할은 (그림 2)의 **Basic_Partition** 알고리즘을 첫번째 차원부터 주기적으로 P개의 MBC를 생성할 때 까지 반복 수행한다. 우리는 데이터가 균등하게 분포한다고 가정하였기 때문에 분할 차원은 주기적으로 선택될 것이며 그리고 만약 어느 한 차원의 하한 경계 부근에서 분할이 발생하면 그 차원의 다음 분할에서는 반드시 상한 경계 부근에서 분할이 발생함을 알 수 있다. 따라서 이차원인 경우 CSP 분할 방법은 (그림 3)과 같은 순서 및 모양으로 이루어 진다.

3.2.2 CSP 방법의 기대값

CSP 분할 결과 생성된 MBC의 각 변의 길이는 한 차원을 제외한 다른 모든 차원들에서는 아주 길다. MBC의 짧은 길이를 가지는 변은 데이터 공간의 경계와 떨어진 거리에 의하여 기대값이 결정됨은 명제 3.1의 (1), (2)에 의하여 알 수 있다. 차원이 높은 경우 작은 선택률이라 하더라도 질의 변의 길이는 1 에 가까운 값이 되고 따라서 $1 - L(q)$ 의 값은 0 에 가까운 값이 된다. 따라서, MBC의 분할되지 않은 d–1개 차원의 변들은 $1 - L(q)$ 보다 큰 값을 가지며 명제 3.1의 (3), (4)에 의하여 이러한 변들의 기대값은 대부분 1이 된다. 이러한 관찰을 보다 자세히 수식적으로 기술함으로써 CSP 방법의 기대값을 알 수 있다. 기대값은 우선 CSP에 의하여 분할된 MBC를 구성하는 범위들의 위치 및 이러한 위치에 의하여 결정되는 거리에 상당히 밀접한 관련이 있다. 우리는 이러한 값을 구하고 난 후, 기대값에 대한 수식을 유도한다.

명제 3.2 d-차원의 단위 하이퍼 공간 $[0,1]^d$이 CSP 방법에 의하여 P개의 MBC들로 분할된다고 하자. 이때, $S_{CSP}(u,v)$는 u번째 분할에 의한 MBC의 v차원의 변의 길이라 하자. 그리고 $L_{CSP}(u,v)$를 v차원의 u번째 분할 이

후 남은 데이터 공간의 v차원의 길이라 하자. 그러면

$$S_{CSP}(u,v) = \frac{1}{P - v + 1}$$

$$S_{CSP}(u,v) = L_{CSP}(u-1,v) \cdot \frac{1}{k}$$

$$L_{CSP}(u,v) = L_{CSP}(u-1,v) \cdot \frac{k-1}{k}$$

이 때, $k = P - d \cdot (u-1) - (v-1)$, $1 \le v \le d$, 그리고 $2 \le u \le \lfloor P/d \rfloor$

증명. [20]을 참조하기 바란다. □

위의 명제 3.2에 의하여 CSP 분할 방법에 의하여 각 차원에서 분할된 길이를 알 수 있다. 매 분할 시 공간을 상호 배타적으로 분할하기 때문에 이전에 분할된 길이를 이용하여 현재 분할된 길이 및 데이터 공간의 양 경계들로부터의 거리 또한 알 수 있다. 다음의 정의는 CSP방법의 기대값을 단순하게 나타내기 위하여 필요한 정의이다.

정의 2 (**누적 길이**, **CL**).

$$CL_{\text{left}}(u,v) = \sum_{i=1}^{u} S_{CSP}(2i-1, v)$$

$$CL_{\text{right}}(u,v) = \sum_{i=1}^{u} S_{CSP}(2i, v)$$

분할된 길이의 누적 길이 또는 데이터 공간의 양 경계들로부터의 거리에 대한 위의 정의를 이용하여 구하고자 하는 기대값을 나타내면 다음의 명제 3.3과 같다.

명제 3.3 CSP 방법에 의하여 분할된 d-차원의 MBC들의 집합 M과 한 변의 길이가 $L(q)$인 질의 입방체 $\bar{\mathbf{q}}$가 주어졌다고 가정하자. 그리고 $L(q) \ge 0.5$일 때 양의 정수 $m1$, $m2$, $v1$, $v2$들은 조건 $CL_{left}(m_1,v_1) \le 1 - L(q)$ 와 $CL_{right}(m_2,v_2) \le 1 - L(q)$ 를 만족하는 최대 정수라 하자. 그러면 $L(q) \ge 0.5$인 경우, 집합 M에서 $\bar{\mathbf{q}}$와 교차하는

MBC들의 기대값 $E_{CSP}(M, \vec{\mathbf{q}})$는 다음과 같다.

$$E_{CSP}(M, \vec{\mathbf{q}}) = \frac{\sum_{i=1}^{m_1-1}\sum_{j=1}^{d} CL_{left}(i,j)}{1-L(q)} + \frac{\sum_{j=1}^{v_1} CL_{left}(m_1,j)}{1-L(q)} + \frac{\sum_{i=1}^{m_2-1}\sum_{j=1}^{d} CL_{right}(i,j)}{1-L(q)} + \frac{\sum_{j=1}^{v_2} CL_{right}(m_2,j)}{1-L(q)} + (P-x) \quad (3)$$

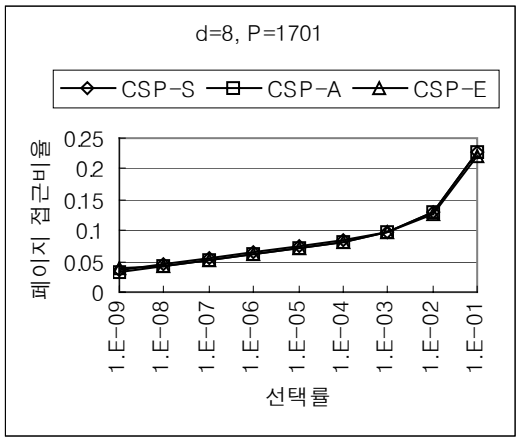
이 때, $x = (m_1 + m_2)\cdot d + v_1 + v_2$ 이고 $P = |M|$ 이다.

증명. [20]을 참조하기 바란다. □

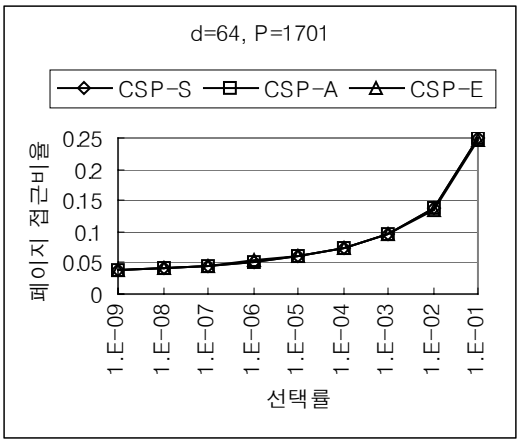
CSP 분할 방법의 기대값은 분할된 MBC를 구성하는 변들의 위치에서 데이터 공간의 경계 사이의 거리를 구한 값들의 합과 데이터 공간의 중앙에 위치한 공간 $[1-L(q), L(q)]^d$와 교차하는 MBC들의 개수로 표현될 수 있다는 사실을 나타낸다. 이러한 사실은 데이터의 개수가 주어진 경우 가능하면 데이터 공간의 경계에서부터 분할하여 패킹을 하는 것이 질의 성능에 효율적이라는 사실을 나타낸다. 그리고, 기존의 균등성을 지향하는 분할 방법은 성능에 부정적인 영향을 미친다는 사실 또한 나타내고 있다. 다음 절에서 우리는 CSP의 성능 모델의 정확성을 가상 및 실제 실험을 통하여 보여준다.

3.2.3 분석의 정확성.

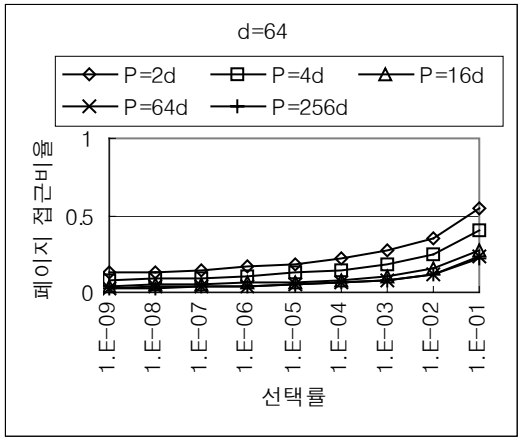
질의 변의 길이 $L(q) \geq 0.5$인 경우, CSP의 기대값은 두 항의 합, 즉 누적길이의 합과 MBC들의 개수로 표현된다. 질의 변의 길이 $L(q) < 0.5$인 경우, 본 연구에서는 기대값에 대한 추정치를 사용한다. 사용된 추정치는 $E_{CSP}(M, \vec{\mathbf{q}}) = \sum\sum CL(i,j)/(1-L(q)) \cdot (1 + (P-x)/P)$이다. 이 값은 누적 길이 조건을 만족하는 MBC들의 기대값의 평균치와 조건을 만족하지 않는 MBC들의 개수(즉, $(P-x)$)의 곱이다. 제안한 CSP의 성능 모델을 다음 두 가지 경우에 대하여 검증한다. 첫째는 CSP 방법으로 $[0,1]^d$ 데이터 공간을 타일링하는 MBC들을 인위적으로 생성하고 질의를 수행한 경우이다(–S). 둘째는 각 차원에 대하여 [0,1] 사이의 수를 랜덤하게 생성하여 *d*-차원의 균등 분포를 가지는 합성된 데이터 집합을 생성하고 이러한 데이터집합을 CSP방법에 따라 분할한 MBC들의



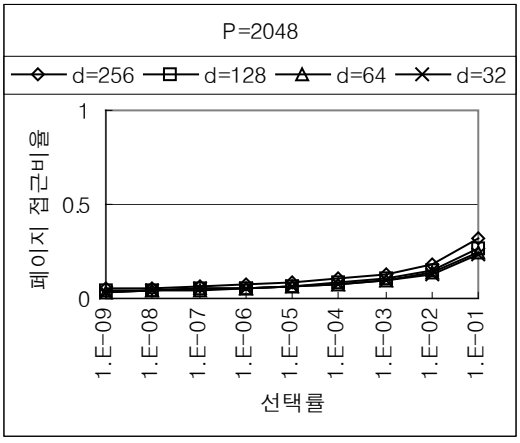
(가) N=68040, 블록킹 인수=40, d=8



(나) N=68040, 블록킹 인수=40, d=64



(다) 고정된 차원에 대한 다양한 P



(라) 고정된 P에 대한 다양한 차원

(그림 4) 분석 결과와 시뮬레이션, 실험과의 비교 그래프

집합에 대하여 질의를 수행한 경우이다(–E). 성능모델을 이용한 기대값은 그래프에서 –A로 표시한다.

실험에서 사용된 선택률의 범위는 10^{-9}에서 10^{-1}까지이다. 예를 들면, 차원이 64인 경우 질의 변의 길이는 0.723에서 0.965까지이다. 데이터의 개수가 68040이고 블록킹 인수가 40인 경우의 실험결과는 (그림 4)의 (가), (나)에서 보여준다. (그림 4)의 (다)는 차원이 일정한 경우 분할 개수 P의 변화에 따른 CSP-A의 디스크 접근 비율을 보여준다. (그림 4)의 (라)는 분할 개수가 일정한 경우 차원의 변화에 따른 CSP-A의 디스크 접근 비율을 보여준다. 디스크 접근 비율은 디스크 접근 횟수를 분할 개수 P로 나눈 값이다. (그림 4)의 (가), (나)에서 우리의 분석 결과가 시뮬레이션 및 실험과 상당히 비슷한 결과를 보여준다. 두 그래프에서 차원이 비록 여덟 배나 증가 하더라도 기대값의 차이는 거의 없음을 보여준다. 따라서, 분할 개수가 동일한 경우 차원의 증가는 페이지 접근 비율적인 면에서 거의 고정적임을 알 수 있다. 이러한 사실은 (그림 4)의 (라)에서 보다 명확하게 보여준다. (그림 4)의 (다)에서 분할 개수 즉 데이터의 개수가 증가하면 오히려 디스크 접근 비율이 어느 정도 까지 감소하는 결과를 보여준다. 이러한 실험 결과를 통하여 우리는 대용량 및 고차원 데이터에 대

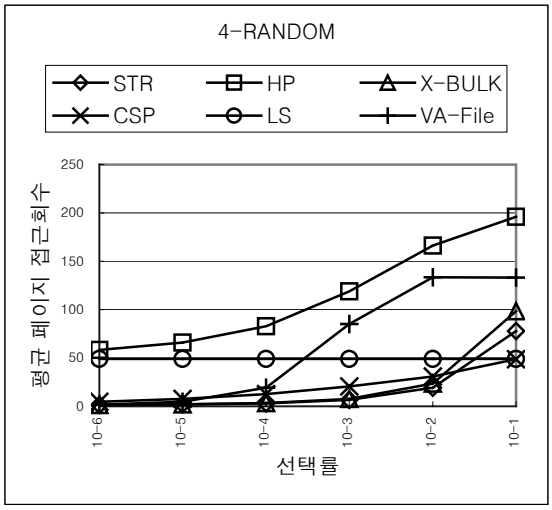
하여 제안한 CSP방법이 디스크 접근 비율적인 면에서 효율적이라는 사실을 알 수 있다.

4. 실험

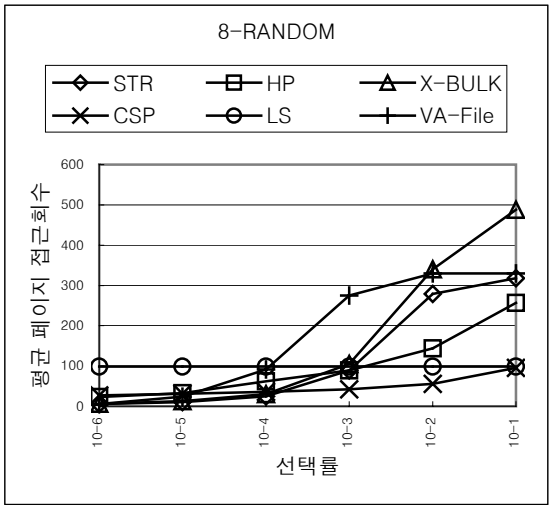
이번 장에서 우리는 CSP와 다른 방법들, 즉, 순차 검색(LS), STR, VA-File, HP 그리고 X-트리와 영역 질의에 대한 디스크 접근 횟수를 비교한다. 순차 검색 방법은 실험의 하한 성능을 나타내는 방법이며 생성된 전체 노드 개수의 1/4만큼의 디스크 접근 회수를 가진다고 가정한다. 균등-깊이 분할방법에 속하는 예로써 STR 과 VA-File 방법을 선택한다. 이 방법들은 이전 장의 분석 결과에 의하여 정방형 질의 $\ddot{\mathbf{q}}$의 한 변의 길이 $L(q)$가 0.5보다 큰 경우에는 항상 CSP 방법에 비하여 나쁜 성능을 보여줄 것이다. STR에서 분할 차원은 일반적으로 사용되고 있는 가장 긴 범위를 가지는 차원을 분할 축으로 선택한다. VA-File의 경우, 각 차원에 할당된 비트 수 b 는 각 데이터에 대하여 적어도 1개의 근사정보를 얻을 수 있는 개수를 선택한다. 즉, 데이터 개수 N 이 주어진 경우 비트 수 b 는 $\geq \lceil \log_2 \lceil N^{1/d} \rceil \rceil$ 이다. VA-파일 같은 경우 페이지에 저장된 데이터들 사이의 거리적 인접성을 보장하지 못하고 질의와 교차하는 근사정보가 여러 페이지에 걸쳐서 존재할 수 있다. 따라서, 우리는 영역 질의를 지원하기 위하여 64개의 페이지들을 할당한 LRU 버퍼를 사용한다. X-트리는 고차원 데이터에 대하여 비교적 뛰어난 성능을 보여주는 동적 색인 방법이다. 따라서, 비교의 공정성을 위하여 우리는 이를 동적으로 구축한 경우보다 성능이 좋은 정적으로 구축한 경우에(X-BULK) 대하여 실험한다. 모든 실험들에서 사용된 페이지의 크기는 8K로 한다.

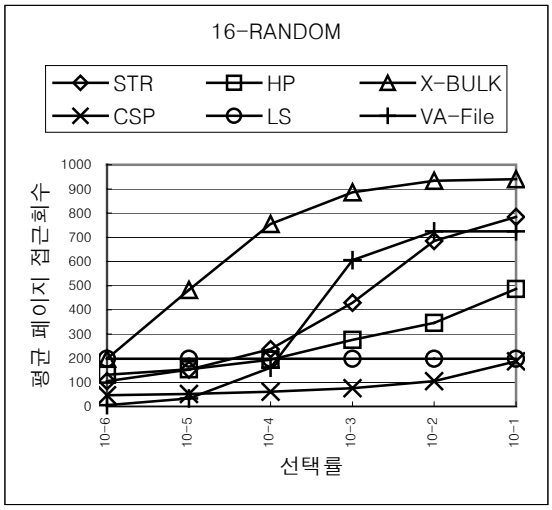
4.1 질의 및 데이터 집합들

실험에 사용된 데이터 집합들은 각각 4, 8, 16 그리고 32차원 집합들이며, 이들 각각은 100,000개의 데이터로 구성된 균등분포를 가지는 데이터 집합이다. 질의 집합은 주어진 데이터 집합으로부터 1000개의 데이터를 임의로 선택한 후, 선택률의 값만큼 범위를 확장하여 생성한다. 선택률은 10^{-6}부터 10^{-1}까지로 한다. 예를 들어, 4차원인 경우 질의 창의 한 변의 길이는 0.03부터 0.56까지 변한다. 차원이 높아지면 한 변의 길이는 증가한다는 사실을 주목하기 바란다.

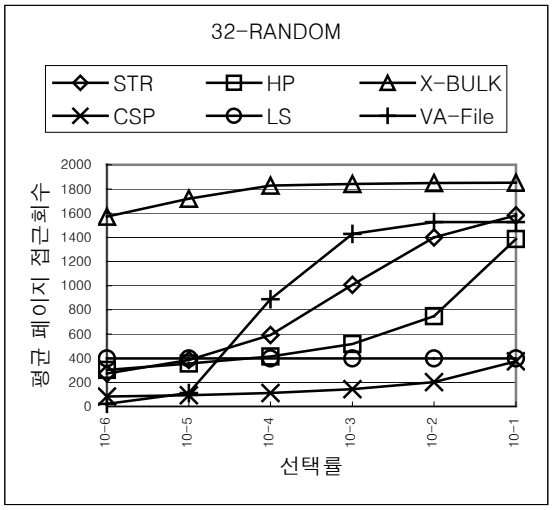


(가) 4-RANDOM





(다) 16-RANDOM



(라) 32-RANDOM

(그림 5) 합성 데이터 집합에 대한 실험 결과

4.2 실험 결과

합성 데이터에 대한 실험 결과는 (그림 5)에서 보여준다. 고차원 색인 방법들 중 비교적 좋은 성능을 보여준다고 지적된 X-트리의 경우 16차원 이상의 데이터 집합에 대해서는 실험에서 사용된 모든 방법들보다 나쁜 성능을 보여주며 오직 차원이 낮은 경우 즉, 4차원인 경우에 한해서 다른 방법들에 비하여 비교적 경쟁력이 있음을 보여준다. 균등 분할 방법의 일종인 STR 과 VA-파일의 경우 모든 차원에 걸쳐서 비슷한 S 형태의 성능을 보여준다. 선택률이 낮은 경우에는 VA-파일의 성능이 STR보다 좋음을 보여준다. 이는 VA-파일에 의하여 분할된 근사화 정보가 차지하는 공간의 크기가 STR에 의하여 분할된 노드의 크기보다 작으며 질의 창

또한 근사화 정보가 차지하는 공간 크기보다 작기 때문이다. 힐버트 공간채움곡선을 이용한 방법은 4차원을 제외한 차원들에서 균등 분할 방법들보다 대체로 좋은 성능을 보여준다는 사실을 주목하기 바란다. CSP 방법을 제외한 방법들은 비교적 낮은 차원과 작은 선택률에서는 순차 검색 방법과 비교하여 좋은 성능을 보여주나 높은 차원과 큰 선택률에서는 순차 검색보다 나쁜 성능을 보여준다. 이러한 현상은 차원이 증가하면 할수록 더욱더 뚜렷해진다. 오직 제안한 CSP 방법만이 모든 차원들 및 선택률들에서 순차 검색 방법보다 좋은 성능을 보여준다. 결론적으로 본 논문에서 제안한 CSP 방법은 차원이 높다 하더라도 좋은 성능을 보여주며 전체 페이지에 대하여 접근하는 페이지 수의 비율은 오히려 차원이 증가하면 감소하는 성질을 보여준다.

5. 결론 및 향후 과제

패킹에서 중요한 결정 사항들 중의 하나는 데이터 분할 방법의 선택이다. 차원이 낮은 경우, 데이터 집합을 균등 크기로 분할하는 것이 좋다. 그러나, 고차원인 경우 한정된 분할 회수 때문에 균등하게 분할하기도 힘들 뿐만 아니라 균등한 분할 방법이 질의 성능에 좋은 영향을 미치지 못한다는 사실을 지적하였다. 따라서, 우리는 민코프스키-합 비용모델에 대한 관찰 결과를 토대로 한 차원에 대해서만 데이터 공간의 경계로부터 박피하듯이 분할하는 불균등 분할 방법이 질의에 긍정적인 영향을 미친다는 사실을 밝혔다. 그리고 이러한 사실을 이용하여 CSP 분할 방법을 제안하고 성능 모델을 제시하였다. 성능 모델의 분석 결과, 한 차원에서의 분할된 범위와 데이터 공간의 양 경계들과의 거리, 그리고 질의 변의 길이에 의하여 구성되는 데이터 공간의 중앙 영역과 겹치는 노드들의 수가 기대값에 직접적으로 영향을 미친다는 사실을 수식을 통하여 보여주었다. 그리고, 시뮬레이션과 합성 데이터에 대한 실험을 통하여 우리는 제안한 수식의 정확성을 보여주었다.

본 논문에서 다양한 차원들 및 선택률들에서 CSP 방법이 다른 방법들보다 뛰어난 영역 질의 성능을 보여줌을 실험을 통하여 확인하였다. 그리고 CSP 방법만이 순차 검색 방법에 비하여 일정한 성능 우위를 보여준다는 사실을 확인하였다.

결론적으로 우리가 제안한 CSP 방법은 다음과 같은 특징이 있다.

(1) 정방형 질의 창 $\bar{\mathbf{q}}$의 한 변의 길이 $L(q)$가 0.5 이상이 되는 고차원 환경에서 유일하게 순차 검색 방법에 비하여 좋은 성능을 보여준다.

(2) 주어진 차원에서 데이터의 개수가 증가할 때, 전체 페이지 개수에 대한 접근 비율은 오히려 감소한다.

(3) 데이터의 개수가 고정된 경우 차원이 증가하더라도 전체 페이지 개수에 대한 접근 비율은 거의 동일하다.

(4) (2), (3)에 의하여 CSP 방법은 차원 및 데이터의 개수의 증가에 덜 민감한 분할 방법이다. 따라서 새로운 응용 환경들에서 요구하는 대용량, 고차원 데이터 처리에 적절한 정적인 분할 방법이라고 주장할 수 있다.

CSP 방법을 디스크배열 등에 적용하는 디클러스터링 방법은 앞으로 우리가 연구할 과제가 될 것이다.

감사의 글

본 연구는 KOSEF F01-2002-000-10014-0, KOSEF R05-2002-000-01288-0 및 지역전략산업 석·박사 연구인력 양성사업에 의하여 지원되었습니다.

참고 문헌

[1] C. Aggarwal, J. Wolf, P. Yu and M. Epelman, “The S-Tree: An Efficient Index for Multidimensional Objects”, *Int. Symp. SSD’97*, page 350~373, 1997.

[2] L. Arge, ‘Efficient External-Memory Data Structures and Applications’, Ph.D. Thesis, *BRICS Dissertation Series, DS–96–03*, University of Aarhus, 1996.

[3] L. Arge, K. Hindrichs, J. Vahrenhold, and J. S. Vitter, "Efficient Bulk Operations on Dynamic R-trees", *ALENEX*, page 328~348, 1999.

[4] D. Barbará, et al., “The New Jersey Data Reduction Report”, *IEEE Bulletin of the Technical Committee on Data Engineering*, 20(4), page 3~45, 1997.

[5] N. Beckmann, H. -P. Kriegel, R. Schneider and B. Seeger, "The R*-tree: An Efficient and Robust Access Method for Points and Rectangles", *Proc. ACM SIGMOD Int. Conf. on Management of Data*, page 322~331, 1990.

[6] R. E. Bellman, ‘Adaptive Control Process’, Princeton University Press, 1961.

[7] S. Berchtold, D. A. Keim and H. -P. Kreigel, "The X-tree: An Index Structure for High-Dimensional Data", *Proc. 22rd Int’l Conf. on Very Large Data Bases* (*VLDB*) page 28~39, 1996.

[8] S. Berchtold, C. Bohm and H. -P. Kriegel, "Improving the Query Performance of High-Dimensional Index Structures by Bulk Load Operations", *Proc. EDBT*, 1998.

[9] J. van den Bercken, B. Seeger and P. W. Widmayer, "A Generic Approach to Bulk Loading Multidimensional Index

Structures", *Proc. 23rd Int'l Conf. on Very Large Data Bases* (*VLDB*) page 406~415, 1997.

[10] C. Böhm, S. Berchtold and D. Keim, "Searching in High-Dimensional Spaces-Index Structures for Improving the Performance of Multimedia Databases", *ACM Computing Surveys*, 33(3), page 322~373, 2001.

[11] S. Chaudhuri and U. Dayal, "An Overview of Data Warehousing and OLAP Technology", *SIGMOD Record*, 1997.

[12] R. Eenk, et al., "Bulk loading a Data Warehouse built upon a UB-Tree", *Proc. IEEE IDEAS*, page 179~187, 2000.

[13] V. Gaede and O. Günther, "Multidimensional Access Methods", *ACM Computing Surveys*, 30(2), page 170~231, 1998.

[14] Y. J. Garcia, M. L. Lopez and S. T. Leutenegger, "A Greedy Algorithm for Bulk Loading R-trees", Technical Report 97–2, 1997.

[15] D. M. Gavrila, "R-tree Index Optimazation", Technical Report CS–TR–3292, 1996.

[16] A. Guttman, "R-trees: A Dynamic Index Structure for Spatial Searching", *Proc. ACM SIGMOD Int. Conf. on Management of Data*. page 47~57, 1984.

[17] G. R. Hjaltason, H. Samet and Y. Sussmann, "Speeding up bulk-loading of quadtrees", *ACM–GIS*. page 50~53, 1997.

[18] A. K. Jain, M. N. Murty and P. J. Flynn, "Data Clustering: A Review", *ACM Computing Surveys*, 31(3), page 264~323, 1999.

[19] I. Kamel and C. Faloutsos, "On Packing R-trees", *Proc. Int. Conf. on Information and Knowledge Management* (*CIKM*), page 490~499, 1993.

[20] T. W. Kim and K. -J. Li, "A Distance-Based Packing Method for High Dimensional Data", *ADC*'03, page 135~144, 2003.

[21] S. T. Leutenegger and D. M. Nicol, "Efficient Bulk-Loading of Gridfiles", ICASE Report 94–74, 1994.

[22] S. T. Leutenegger, M. A. Lopez and J. Edington, "STR: A Simple and Efficient Algorithm for R-Tree Packing", *Proc. 13th Int. Conf. on Data Engineering* (*ICDE*), 1997.

[23] S. T. Leutenegger and M. A. Lopez, "The Effect of Buffering on the Performance of R-Trees", *Proc. 14th Int. Conf. on Data Engineering* (*ICDE*). page 164~171, 1998.

[24] B. -U. Pagel, H. -W. Six, H. Toben and P. W. Widmayer, "Towards an Analysis of Range Query Performance in Spatial Data Structures", *ACM PODS*, 1993.

[25] B. -U. Pagel, H. -W. Six and M. Winter, "Window Query-Optimal Clustering of Spatial Objects". *ACM PODS*, page 86~94, 1995.

[26] J. T. Roussopoulos and L. Leifker, "Direct spatial search on pictorial databases using r-trees", *Proc. ACM SIGMOD Int. Conf. on Management of Data*, 1985.

[27] D. White and R. Jain, "Similarity Indexing: Algorithms and Performance", *Int. Symp. on Optical Science and Technology* (*SPIE*), page 62~73, 1996.

[28] R. Wober, H. -J. Schek and S. Blott, “A Quantitative Analysis and Performance Study for Similarity-Search Methods in High-Dimensional Spaces”, *Proc. 24th Int’l Conf. on Very Large Data Bases*(VLDB), page 194~205, 1998.