이원성 기반 시계열 서브시퀀스 매칭의 인덱스 검색을 위한 최적의 기법

김 상 욱[†] · 박 대 현[††] · 이 헌 길[††]

요 약

본 논문에서는 시계열 데이터베이스에서 서브시퀀스 매칭을 효과적으로 처리하는 방안에 관하여 논의한다. 먼저, 본 논문에서는 서브시퀀스 매칭을 위한 기존 기법의 인덱스 검색에서 발생하는 성능상의 문제점들을 지적하고, 이들을 해결할 수 있는 새로운 방법을 제시한다. 제안된 기법은 서브시퀀스 매칭의 인덱스 검색 문제를 *윈도우-조인*이라는 일종의 공간 조인 문제로 새롭게 해석하는 것에서 출발한다. 윈도우-조인의 빠른 처리를 위하여 제안된 기법에서는 서브시퀀스 매칭을 시작할 때 질의 시퀀스를 위한 R*-트리를 주기억장치 내에 구성한다. 또한, 제안된 기법은 데이터 시퀀스들을 위한 디스크 상의 R*-트리와 질의 시퀀스를 위한 주기억장치 상의 R*-트리를 효과적으로 조인할 수 있는 새로운 알고리즘을 포함한다. 이 알고리즘은 데이터 시퀀스들을 위한 R*-트리 페이지들을 *인덱스 단계의 착오 채택* 없이 단 한번만 디스크로부터 액세스하므로 디스크 액세스 측면에서 최적의 기법임이 증명된다. 또한, 다양한 실험을 통한 성능 평가를 통하여 제안된 기법의 우수성을 정량적으로 규명한다.

An Optimal Way to Index Searching of Duality-Based Time-Series Subsequence Matching

Sang-Wook Kim[†] · Dae-Hyun Park[††] · Heon-Gil Lee[††]

ABSTRACT

In this paper, we address efficient processing of subsequence matching in time-series databases. We first point out the performance problems occurring in the index searching of a prior method for subsequence matching. Then, we propose a new method that resolves these problems. Our method starts with viewing the index searching of subsequence matching from a new angle, thereby regarding it as a kind of a spatial-join called a *window-join*. For speeding up the window-join, our method builds an R*-tree in main memory for a query sequence at starting of subsequence matching. Our method also includes a novel algorithm for joining effectively one R*-tree in disk, which is for data sequences, and another R*-tree in main memory, which is for a query sequence. This algorithm accesses each R*-tree page built on data sequences *exactly once* without incurring any *index-level false alarms*. Therefore, in terms of the number of disk accesses, the proposed algorithm proves to be optimal. Also, performance evaluation through extensive experiments shows the superiority of our method quantitatively.

키워드 : 시계열 데이터베이스(Time-Series Databases), 서브시퀀스 매칭(Subsequence Matching), 인덱스 검색(Index Searching), 윈도우-조인(Window-Join)

1. 서 론

시계열 데이터베이스(time-series databases)란 각 객체의 변화되는 값들의 연속으로 구성된 데이터 시퀀스(data sequences)들의 집합이다. 유사 검색(similarity search)이란 주어진 질의 시퀀스(query sequence)와 변화의 추세가 유사한 시퀀스 혹은 서브시퀀스들을 시계열 데이터베이스로부터 검색하는 연산이다[1, 5]. 유사 검색은 데이터 마이닝(data mining) 및 데이터 웨어하우징(data warehousing) 분야에서 중요한 연산으로 사용된다[7, 9, 11].

유사 검색은 크게 전체 매칭(whole matching)과 서브시퀀스 매칭(subsequence matching)으로 분류된다[1, 5, 7]. 전체 매칭은 모든 데이터 시퀀스들과 질의 시퀀스의 길이가 동일하다는 전제 하에 질의 시퀀스와 유사한 시퀀스를 데이터베이스로부터 찾아낸다. 서브시퀀스 매칭은 임의의 길이의 질의 시퀀스와 유사한 데이터 시퀀스 내의 서브시퀀스를 찾아낸다. 이와 같이, 서브시퀀스 매칭은 길이에 대한 제약이 없으므로 다양한 실제 응용 분야에서 널리 사용된다[10].

※ 본 연구는 2003년도 한양대학교 교내 연구비(HY-2003) 지원과 정보통신부 IT 연구센터(강원대학교 미디어서비스기술 연구센터)의 연구비 지원으로 수행되었습니다.
† 종신회원 : 한양대학교 정보통신학부
†† 정 회 원 : 강원대학교 컴퓨터정보통신공학부
논문접수 : 2004년 5월 6일, 심사완료 : 2004년 7월 27일

본 논문에서는 시계열 데이터베이스에서 서브시퀀스 매칭을 효과적으로 처리하는 방안에 관하여 논의하고자 한다. 먼저, 본 논문에서는 Dual-Match라 부르는 서브시퀀스 매칭을 위한 기존 기법의 인덱스 검색에서 발생하는 성능상의 문제점들을 지적하고, 이들을 해결할 수 있는 새로운 방법을 제시한다. 제안된 기법은 서브시퀀스 매칭의 인덱스 검색 문제를 윈도우-조인(window-join)이라는 일종의 공간 조인 문제(spatial-join)[3, 6, 12]로 새롭게 해석하는 시도에서 출발한다. 윈도우-조인의 빠른 처리를 위하여 제안된 기법에서는 서브시퀀스 매칭을 시작할 때 질의 시퀀스로부터 추출된 질의 윈도우 점들을 대상으로 R^*-트리를 생성한다. 또한, 제안된 기법은 두개의 R^*-트리를 이용하여 윈도우-조인을 효율적으로 처리하기 위한 새로운 알고리즘을 포함한다. 여기서, 하나의 R^*-트리는 전술한 질의 윈도우 점들을 대상으로 주기억장치 내에 존재하는 것이며, 또 다른 하나의 R^*-트리는 데이터 윈도우 점들을 대상으로 디스크 내에 존재하는 것이다. 제안된 윈도우-조인 알고리즘이 디스크 액세스 측면에서 최적을 보장함을 증명한다. 또한, 다양한 실험을 통한 성능 평가를 통하여 제안된 기법의 우수성을 정량적으로 규명한다.

본 논문의 구성은 다음과 같다. 제2장에서는 시계열 서브시퀀스 매칭과 연관된 기존의 기법들에 관하여 요약한다. 제3장에서는 본 연구의 동기가 되는 기존 기법들의 공통적인 문제점들을 지적한다. 제4장에서는 이러한 문제점들을 해결하는 새로운 기법을 제안한다. 제5장에서는 제안된 기법의 우수성을 규명하기 위한 성능 평가 결과를 제시한다. 끝으로, 제6장에서는 결론을 내리고 향후 연구 방향을 제시한다.

2. 관련 연구

본 장에서는 본 논문에서 해결하고자 하는 문제를 정의하고, 이 문제에 대한 해결 방안을 제시한 기존의 연구 결과들을 요약한다.

2.1 문제 정의

시계열 데이터베이스 D, 길이 n의 질의 시퀀스 $Q=(q[0], q[1], \cdots, q[n-1])$, 그리고 유사 허용치 ε이 주어질 때, 서브시퀀스 매칭 문제는 다음과 같이 정의된다. D에 저장된 길이 N의 임의의 시퀀스 $S=(s[0], s[1], \cdots, s[N-1])$ 내에 존재하는 길이 n의 임의의 서브시퀀스 $X=(x[0], x[1], \cdots, x[n-1])$가 다음 조건을 만족하면, 〈S, S내 X의 오프셋〉을 반환한다.

$$d(X,Q) \le \varepsilon, \text{ 여기서 } d(X,Q) = \sqrt{\sum_{i=0}^{n-1} (x[i]-q[i])^2} \text{ 1)}$$

1) 즉, 유클리드 거리(Euclidean distance)가 주어진 허용치 ε이하인 두 서브시퀀스 X와 Q를 유사하다고 간주한다.

2.2 FRM

참고문헌 [5]에서는 서브시퀀스 매칭을 위한 좋은 해결 방안을 제안하였다. 본 논문에서는 참고 문헌 [10]의 명칭을 따라 이 기법을 FRM이라 부른다. FRM에서는 미리 고정된 크기 w를 갖는 윈도우(window)라는 개념을 이용한다.

먼저, 각 데이터 시퀀스의 모든 가능한 위치로부터 길이 w의 슬라이딩 윈도우(sliding window)들을 추출하고, 각 슬라이딩 윈도우를 이산 퓨리에 변환(discrete Fourier transform : DFT)을 이용하여 저차원 공간상의 점으로 변환한다. 본 논문에서는 이후부터 이 점을 데이터 윈도우 점(data window point)이라 정의한다. 이러한 방식으로 생성되는 데이터 윈도우 점들의 수가 매우 많으므로, FRM에서는 다수의 데이터 윈도우 점들을 포함하는 최소 포함 사각형(minimum bounding rectangle : MBR)들을 형성한 후, 이 MBR들을 다차원 인덱스(multidimensional index)의 하나인 R^*-트리[2]에 저장한다.

서브시퀀스 매칭을 위하여 질의 시퀀스로부터 크기 w의 디스조인트 윈도우(disjoint window)들을 추출하고, 이러한 디스조인트 윈도우들을 DFT하여 저차원 공간상의 윈도우 점들로 변환한다. 본 논문에서는 이후부터 이러한 점을 질의 윈도우 점(query window point)이라 정의한다. 각 질의 윈도우 점에 대하여 $\varepsilon/\sqrt{p}$을 유사 허용치로 갖는 범위 질의를 R^*-트리 상에서 수행한다. 여기서, p=⌊(질의 시퀀스 길이)/w⌋이다. 이러한 범위 질의의 결과로 얻어진 데이터 윈도우 점들을 조사함으로써 최종 결과에 포함될 가능성이 높은 후보 서브시퀀스(candidate subsequence)들을 파악한다. 다음으로 이 후보 서브시퀀스들을 포함하는 각 시퀀스를 디스크로부터 액세스하여 후보 서브시퀀스와 질의 시퀀스와의 유클리드 거리를 실제로 계산한다. 끝으로, 이러한 계산 결과 착오 채택(false alarm)[1]으로 밝혀진 것들을 제외한 나머지 서브시퀀스들을 최종 결과로 반환한다.

2.3 Dual-Match

FRM에서는 인덱싱을 위한 저장 공간의 오버헤드를 줄이기 위하여 개별적 윈도우 점들 대신 다수의 윈도우 점들을 포함하는 MBR들을 R^*-트리 내에 저장한다. 그러나 이러한 MBR 내부에는 죽은 공간(dead space)[2]이 존재하게 되므로, 이로 인하여 후보 서브시퀀스의 착오 채택이 발생되며, 이것은 서브시퀀스 매칭의 성능 저하로 직결된다[10]. 참고 문헌 [10]에서는 이러한 문제점을 해결하기 위한 방법으로서 이원성 기반 서브시퀀스 매칭(duality-based subsequence matching : Dual-Match)을 제안하였다.

데이터 시퀀스로부터 슬라이딩 윈도우를 추출하고 질의 시퀀스로부터 디스조인트 윈도우를 추출하는 FRM과는 반대로 Dual-Match에서는 데이터 시퀀스로부터는 디스조인트 윈도우를 추출하고 질의 시퀀스로부터는 슬라이딩 윈도우를 추출

하는 방식을 사용한다. 이와 같은 단순한 역할 교환을 통하여 Dual-Match에서는 R*-트리에 저장할 데이터 윈도우들의 수를 FRM의 약 1/w로 줄일 수 있다. 이 결과, MBR들을 저장하는 FRM과는 달리 Dual-Match에서는 윈도우 점 자체를 R*-트리에 저장하는 것이 가능해 진다. 따라서 Dual-Match에서는 MBR 내의 죽은 공간으로 인한 성능 저하의 문제가 발생되지 않으므로, 전체 서브시퀀스 매칭의 성능이 크게 개선된다.

3. 연구 동기

Dual-Match에서는 질의 시퀀스로부터 슬라이딩 윈도우를 추출하므로, FRM에서와는 달리 매우 많은 수의 질의 윈도우 점들이 생성된다. 예를 들어, 참고 문헌 [10]의 실험에서 사용한 것과 같이 질의 시퀀스 길이가 1024이고 윈도우 길이가 256인 경우에 생성되는 윈도우 점들의 수는 769(= 1,024-256+1)개이다. 본 연구는 이와 같이 많은 질의 윈도우 점들이 생성됨으로 인한 Dual-Match의 성능상의 문제점을 파악하는 것으로부터 출발하였다[2].

참고문헌 [10]에서는 다수의 질의 윈도우 점들이 발생하는 경우 R*-트리를 검색하는 세 가지 방법을 제시하고 있다. 본 장에서는 본 연구의 동기로서 각 방법에서 나타나는 성능상의 문제점들을 각각 지적한다.

- **방법 (1)** : 발생한 각 질의 윈도우 점에 대하여 개별적인 영역 질의를 수행

이 방법에서는 윈도우 점들의 개수만큼 R*-트리를 검색해야 한다. 위 예의 경우, 이것은 R*-트리가 총 769회 검색되어야 함을 의미한다. 특히, R*-트리 공간상에서 인접한 질의 윈도우 점들에 대한 R*-트리 검색은 동일한 R*-트리 페이지를 디스크로부터 여러 번 액세스하도록 하는 결과를 초래하게 된다. 따라서 이러한 R*-트리 검색 비용은 매우 심각하다.

- **방법 (2)** : 모든 질의 윈도우 점들을 포함하는 단 하나의 대형 질의-MBR에 대하여 단 하나의 영역 질의를 수행

이것이 참고문헌 [10]에서 실제로 선택한 방법이다. 이 방법에서는 질의 시퀀스의 길이와 윈도우 크기의 관계없이 단 한번의 R*-트리 검색만이 발생하게 된다. 따라서 방법 (1)에서와 같이 동일한 R*-트리 페이지를 다수 액세스하는 문제는 발생하지 않는다. 반면, 이 방식에서는 영역 질의에서 사용되는 MBR(간략히, **질의-MBR**) 내에 존재하는 죽은 영역으로 인한 후보 서브시퀀스의 착오 채택이 발생할 수 있다. 그러나 Dual-Match에서는 서브시퀀스 매칭의 처리 시간 중 모든 질의 윈도우 점들이 주기억장치 내에서 관리된다는 점을 활용하여 이 문제를 완화시킨다[10]. 즉, R*-트리 검색을 통하여 후보로 반환되는 각 데이터 윈도우 점이 실제로 적어도 한 질의 윈도우 점과 $\varepsilon/\sqrt{p}$ 이하의 거리 이내에 존재하는가를 R*-트리 검색 단계에서 직접 확인하는 것이다. 참고문헌 [10]에서는 이러한 과정을 인덱스 수준 여과(index-level filtering)라 정의한다. 이 결과, 방법 (1)과 방법 (2)에서 R*-트리 검색 결과로 반환되는 후보 서브시퀀스들의 수는 차이가 없게 된다.

그러나 이 방법은 다음과 같은 두 가지 문제점을 가진다.

첫째, 방법 (1)에서는 액세스되지 않던 R*-트리 페이지들을 추가로 액세스하는 경우가 발생한다. 이것은 질의-MBR 내에 존재하는 죽은 공간으로 인한 것이다. 이 죽은 공간은 그 어떤 질의 윈도우 점과도 $\varepsilon/\sqrt{p}$ 이상의 거리에 존재하는 R*-트리 내의 MBR들을 $\varepsilon/\sqrt{p}$ 이내의 거리에 있다고 잘못 판단하도록 하고, 이 결과, 불필요한 R*-트리 페이지들을 추가로 디스크로부터 액세스한다. 본 논문에서는 이러한 문제를 **인덱스 단계의 착오 채택(index-level false alarm)**이라 정의한다. 이러한 문제가 발생하는 근본적인 원인은 질의-MBR 내의 죽은 영역으로 인한 것이다. 모든 질의 윈도우 점들을 포함하는 하나의 질의-MBR은 매우 크므로 그 내부의 죽은 영역 또한 매우 크다. 따라서 인덱스 단계의 착오 채택으로 인한 추가적인 인덱스 페이지 액세스의 오버헤드는 심각하다.

둘째, 인덱스 수준 여과[10]에서 발생하는 CPU 오버헤드이다. 질의-MBR에 의하여 반환되는 **각각의** 후보 데이터 윈도우 점은 **모든** 질의 윈도우 점들과의 비교를 수행하는 인덱스 수준 여과를 거쳐야 한다. 앞의 예에서의 경우, 각 후보 윈도우 점은 769개의 질의 윈도우 점들과의 비교 연산을 수행해야 함을 의미한다. 큰 질의-MBR에 의하여 반환되는 후보 윈도우 점들의 수는 매우 많으므로, 이러한 CPU 오버헤드 역시 전체 서브시퀀스 매칭의 성능을 떨어뜨리는 중요한 원인이 된다.

- **방법 (3)** : 다수의 질의 윈도우 점들을 포함하는 다수의 질의-MBR들에 대하여 각각 영역 질의를 수행

이 방법은 방법 (1)과 방법 (2)의 절충안이다. R*-트리의 검색 회수는 형성되는 질의-MBR들의 수와 동일하다. 이때 가장 자연스러운 의문은 성능이 극대화 할 수 있도록 질의-MBR들을 어떻게 최적으로 형성시킬 것인가 하는 것이다. 질의-MBR을 생성하는 방법으로는 모든 질의-MBR이 같은 수의 윈도우 점들을 포함하도록 하는 방법과 질의-MBR의 수 및 죽은 영역의 크기를 동시에 최소화하는 휴리스틱 방법 등 참고문헌 [5]에서 제시한 방식을 고려할 수 있다.

이 방법은 두 극단적인 방법 (1)과 방법 (2)의 중앙에 위치

2) 본 장에서 지적하는 대부분의 현상이 FRM에서도 동일하게 발생한다. 그러나 FRM과 비교하여 Dual-Match에서 생성되는 질의 윈도우 점들의 수가 훨씬 많으므로 큰 성능상의 문제로 부각되는 것이다.

하는 방법이다. 이 결과, 방법 (1)과 방법 (2)에서 발생하는 각 문제점을 완화시키는 경향이 있는 반면, 각 방법이 가지는 장점을 희석시키는 효과를 갖는다. 따라서 근원적인 해결책으로 사용하기에는 어려움이 있다.

4. 제안하는 기법

본 장에서는 제3장에서 지적한 문제점들을 해결할 수 있는 새로운 서브시퀀스 매칭 기법을 제안한다. 제안하는 기법은 Dual-Match를 기반으로 한다. 따라서 데이터 시퀀스들로부터 디스조인트 윈도우들을 추출함으로써 R^*-트리 인덱스를 구성하고, 서브시퀀스 매칭의 처리 시에는 질의 시퀀스로부터 슬라이딩 윈도우들을 추출한다. 제안하는 기법에서 추구하는 궁극적인 목표는 서브시퀀스 매칭 시 발생하는 디스크 액세스 횟수와 CPU 비용을 최소화하는 것이다.

4.1 문제의 재해석

본 논문에서는 서브시퀀스 매칭에서 후보 데이터 윈도우를 찾는 과정을 아래와 같이 윈도우-조인(window-join) 문제로 재해석한다.

[정의 1] 윈도우-조인(window-join)

데이터 윈도우 점들의 집합 D_W와 질의 윈도우 점들의 집합 Q_W를 대상으로 상호 $\varepsilon/\sqrt{p}$ 이하의 거리를 가지는 <데이터 윈도우 점, 질의 윈도우 점>의 쌍들의 집합을 찾는 공간 조인 문제. □

서브시퀀스 매칭에 관한 기존의 다양한 연구가 있었으나, 저자들이 아는 한 이와 같이 후보 윈도우들을 찾는 문제를 공간 조인(spatial join)[3, 6, 12]의 관점에서 해석한 시도는 없었다. 이러한 해석이 가지는 중요한 의미는 이 문제에 대한 새로운 관점에서의 해결책을 마련할 수 있다는 데 있다. 본 연구에서는 이러한 점에 착안하여 D_W와 Q_W간의 공간 조인을 효과적으로 수행하는 새로운 기법을 고안하고자 한다.

4.2 윈도우-조인의 처리 방안

GIS 및 공간 데이터베이스 분야에서 다루어진 가장 중요한 문제의 하나는 두 개의 데이터 집합에 대한 공간 조인을 효과적으로 처리하는 것이다[3, 6]. 현재까지 알려진 가장 효과적인 방법 중의 하나는 대상이 되는 두 집합에 R^*-트리와 같은 다차원 인덱스가 존재한다는 가정 하에 인덱스 기반 공간 조인(index-based spatial join)을 수행하도록 하는 것이다[3, 6, 12].

그러나 현재 우리가 직면한 윈도우-조인 문제에서는 D_w에는 R^*-트리가 존재하지만, Q_w에는 R^*-트리가 존재하지 않는다. 따라서 참고문헌 [3, 6]에서 제시한 공간 조인 알고리즘을 직접 적용하기에는 어려움이 있다. 본 연구에서는 이를 극복하기 위한 방안으로서 Q_w를 위한 R^*-트리를 서브시퀀스 매칭의 시작 시점에 주기억장치 내에 즉석으로 생성하는 방법을 사용한다. 물론, R^*-트리의 즉석 생성은 서브시퀀스 매칭의 처리를 위한 추가 비용을 요구한다. 그러나 이러한 비용은 다음 세 가지 사실들을 고려할 때 상대적으로 미미하다 : (1) Q_w가 D_w에 비하여 훨씬 작다, (2) Q_w내의 윈도우 점들은 이미 주기억장치 내에 존재한다, (3) Q_w를 위한 R^*-트리는 주기억장치 내에 생성된다. 또한, 전체 윈도우-조인 처리 시 발생하는 디스크 액세스를 크게 줄일 수 있으므로, Q_w를 위한 R^*-트리의 즉석 생성은 정당화 될 수 있다. R^*-트리의 즉석 생성은 질의 윈도우 점들을 반복적으로 주기억장치 내의 R^*-트리 내에 삽입함으로써 가능하다. R^*-트리의 생성 효율을 극대화하기 위하여 벌크 로딩 알고리즘(bulk loading algorithm)들을 활용할 수도 있다.

이제 D_w와 Q_w에 모두 R^*-트리가 존재하므로, 참고 문헌 [3, 6] 등에 제안된 기존의 기법을 활용함으로써 공간 조인을 효과적으로 처리할 수 있다. 그러나 기존의 공간 조인 기법들은 두 R^*-트리 모두가 디스크 상에 존재한다는 것을 가정한다. 따라서 두 R^*-트리의 중요도를 동등하게 취급함으로써 두 R^*-트리에서 발생하는 디스크 액세스 수를 동시에 최적화 하려고 시도한다. 그러나 전체 공간 내의 점들은 다양한 형태로 분포하므로, 이러한 시도의 결과 동일한 인덱스 페이지가 두 번 이상 액세스되는 현상이 발생된다[3, 6].

반면, 현재 우리의 윈도우-조인에서는 D_w를 위한 R^*-트리는 디스크 내에, 그리고 Q_w를 위한 R^*-트리는 주기억장치 내에 존재한다. 따라서 본 연구에서는 윈도우-조인의 고유한 특징을 활용함으로써 최적의 성능을 제공하는 새로운 공간 조인 기법을 제시할 수 있다.

제안하는 공간 조인 기법에서는 D_W를 위한 디스크 상의 R^*-트리 인덱스 I_D를 외부 릴레이션(outer relation), Q_w를 위한 주기억장치 상의 인덱스 I_Q를 내부 릴레이션(inner relation)으로 간주한다. 제안하는 공간 조인 기법의 수행 과정을 간략히 요약하면 다음과 같다. (1) I_D의 루트 페이지 내에 있는 각 MBR에 대하여 이를 $\varepsilon/\sqrt{p}$ 만큼 확장한다. 이 MBR을 e-MBR(enlarged MBR)이라 부른다. (2) e-MBR을 사용하여 I_Q에 영역 질의를 수행한다. (3) 영역 질의의 결과, I_Q내의 그 어떤 질의 윈도우 점과도 $\varepsilon/\sqrt{p}$ 이상 떨어진 것이 확인된 I_D의 MBR에 대해서는 그 하위 서브트리를 이후의 고려 대상에서 완전히 제외시킨다. 이것은 이 MBR이 I_Q내의 어떠한 질의 윈도우 점과도 $\varepsilon/\sqrt{p}$ 이상의 거리에 있음을 의미한다. (4) 반면, 영역 질의의 결과, I_Q내의 어떤 질의 윈도우 점과 $\varepsilon/\sqrt{p}$ 이내에 있다고 판단된 MBR인 c-MBR(confirmed MBR)에 대해서는 I_D내의 그 하위 단계 페이지에 대하여 이러한 작업을 재귀적으로 반복한다.

제안된 기법에서는 I_D와 I_Q모두에 대하여 깊이 우선 탐색(depth first search) 방식으로 윈도우-조인을 처리한다. 또한, 각 e-MBR을 이용하여 I_Q에 대한 영역 질의를 수행할 때, CPU-최적화를 위하여 다음과 같은 방식을 사용한다. 영역 질의 수행 중, e-MBR이 I_Q의 어떤 MBR을 완전히 포함하는 경우, 즉시 진행을 중단하고 이 e-MBR을 c-MBR로 간주한 후, 리턴한다. I_D의 리프 페이지를 액세스한 경우에는 이와 대응되는 e-MBR에 대한 영역 질의 결과 반환된 I_Q의 윈도우 점들과 이 페이지 내의 윈도우 점들을 조인함으로써 최종 후보 윈도우 쌍들의 집합을 구하게 된다.

(알고리즘 1)은 제안된 기법에 의하여 서브시퀀스 매칭을 처리하는 전체 과정을 스케치한 것이다. 페이지 크기의 주기억장치 영역 P_D는 재귀 함수 Window_Join의 입력 파라메터이다. I_D의 리프가 아닌 페이지 내의 엔트리 e는 두 개의 컴포넌트 e.MBR과 e.PID를 가진다. e.MBR은 R*-트리 내에서 루트 페이지 식별자가 e.PID인 서브 트리 내에 저장된 모든 데이터 윈도우들을 포함하는 MBR을 표현한다.

(알고리즘 1)은 Main 함수와 Window_Join 함수의 두 부분으로 구성된다. Main 함수는 먼저 I_D의 루트 페이지의 내용을 디스크로부터 읽은 후, 이를 P_D내에 저장한다. 그 다음, Window_Join 함수를 재귀적으로 호출한다. Window_Join 함수가 종료되면, Main 함수는 Window_Join 함수에 의하여 반환된 모든 후보 서브시퀀스를 조사하여 실제 질의 시퀀스와의 유클리드 거리가 ε이하인 서브시퀀스들만을 최종 결과로 반환시킨다. Window_Join 함수는 재귀적으로 수행되며, 적어도 한 질의 윈도우와의 유클리드 거리가 $\varepsilon/\sqrt{p}$이하인 모든 후보 데이터 윈도우들을 찾아낸다.

```
Main
  1. read the root page of ID from the disk.
  2. store it into P_D, a page-sized memory chunk.
  3. call Window_Join(P_D) to get the candidate set.
  4. FOR every subsequence in the candidate set obtained in Step 3,
    4.1. access the sequence containing the candidate subsequence
         from the disk.
    4.2. calculate the Euclidean distance between the sub-
         sequence and the query sequence.
    4.3. IF the distance is smaller than ε, THEN return the sub-
         sequence as one of the final query results.
End_Function
```

(a) Main 함수

```
Window_Join(PD)
  1. IF PD is a leaf page of ID, THEN
    1.1. join a set of data window points in PD and a set of
         query window points in IQ in a nested-loop fashion.
    1.2. output a set of pairs <data window point, query window
         point> as a join result.
    1.3. RETURN.
    End-IF
  2. ELSE IF PD is a non-leaf page of ID, THEN FOR each en-
     try e in PD,
    2.1. make e-MBR by enlarging e.MBR by ε/√p
    2.2. perform a range query with this e-MBR on I_Q
    2.3. IF the range query returns a non-empty set as a result,
         THEN
      2.3.1. read the next lower-level page pointed by e.PID
             from the disk.
      2.3.2. store it into Local_PD, a page-sized memory chunk.
      2.3.3. call Window_Join(Local_PD).
      End_IF
    End_ELSE_IF
End_Window_Join
```

(b) Window-Join 함수

(알고리즘 1) 서브시퀀스 매칭의 처리 알고리즘

4.3 논의 사항

제안된 기법의 견고성(robustness)은 다음의 Lemma 1에 의하여 규명된다.

[Lemma 1] 윈도우-조인을 기반으로 하는 제안된 기법은 시계열 데이터베이스를 위한 서브시퀀스 매칭을 처리할 때 착오 기각을 유발시키지 않는다.

[Proof] Dual-Match는 서브시퀀스 매칭의 처리시 착오 기각을 유발 시키지 않는 것으로 증명된 바 있다[10]. 또한, 그 정의에 의하여 윈도우-조인은 상호 유클리드 거리가 $\varepsilon/\sqrt{p}$이하인 모든 <데이터 윈도우 점, 질의 윈도우 점>의 쌍들을 성공적으로 찾아낸다. 따라서 이 윈도우 조인의 처리 결과로 반환되는 집합은 원래의 Dual-Match의 인덱스 검색 단계에서 반환되는 집합과 정확히 일치한다. 제4.2절에서 언급한 바와 같이, 제안된 기법은 Dual-Match와 동일한 후처리 방식을 사용한다. 따라서 제안된 기법에 의한 서브시퀀스 매칭의 처리 결과는 항상 Dual-Match에 의한 처리 결과와 동일하며, 이 결과 [Lemma 1]은 증명된다.

5. 성능 평가

본 장에서는 다양한 실험을 통한 성능 평가에 의하여 제안된 기법의 우수성을 규명한다.

5.1 제안된 기법의 성능에 관한 논의

참고문헌 [3, 6]의 기법들에서와는 달리, 제안된 기법에서는 ID를 외부 릴레이션으로 간주하므로 윈도우-조인의 처리 시 액세스되는 ID를 위한 각 R*-트리 페이지는 디스크로부터 단 한번만 액세스된다. 따라서 제3장에서 언급한 Dual-Match의 인덱스 검색 방법 (1)과 (3)에서와 같은 동일한 R*-트리 페이지를 두 번 이상 액세스하는 현상은 발생하지 않는다.

또한, 윈도우-조인의 처리 시, I_Q에 대한 영역 질의를 통하여 액세스가 반드시 필요하다고 판단되는 I_D내의 R*-트리 페이지만을 디스크로부터 액세스한다. 이 결과, 제안된 기법

에서는 인덱스 단계의 착오 채택이 전혀 발생하지 않는다. 따라서 제3장에서 언급한 Dual-Match의 인덱스 검색 방법 (2)와 (3)에서 발생하는 인덱스 단계 착오 채택으로 인한 디스크 액세스의 오버헤드가 제거된다.

끝으로, 제안된 기법에서는 인덱스 여과 과정에서 발생하는 윈도우 점들 간의 비교 횟수를 크게 줄일 수 있다. Dual-Match의 인덱스 검색 방법 (2)와 방법 (3)에서는 각 후보 데이터 윈도우 점을 모든 질의 윈도우 점들과 비교해야 한다. 반면, 제안하는 윈도우-조인 기법에서 각 후보 윈도우 점은 이를 포함하는 상위 단계 e-MBR을 이용한 영역 질의 결과로 반환되는 질의 윈도우 점들만을 비교 대상으로 한다. 따라서 제안된 기법의 CPU 성능은 크게 개선된다.

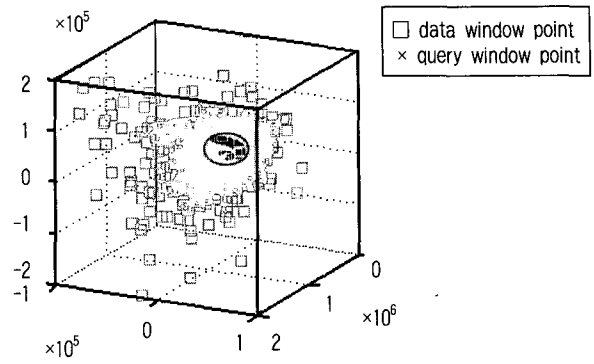
제안된 기법에서는 (1) I_Q를 위한 R*-트리의 즉석 구성 및 (2) I_D에 대한 e-MBR들을 이용한 I_D에 대한 영역 질의 등과 같은 추가 작업이 요구된다. 그러나 이 두 가지 작업들은 모두 주기억장치 내의 데이터를 대상으로 수행되며, 또한, 제4.2절에서 제안된 알고리즘에 의하여 최적화된다. 따라서 이러한 두 가지 추가 작업으로 인한 오버헤드는 위에서 언급한 성능 개선 효과와 비교하면 미미하다.

5.2. 실험 결과 및 분석

먼저, 본 연구에서는 윈도우-조인의 처리 과정에서 I_D내의 얼마나 많은 R*-트리 페이지들이 상위 단계에서 cut-off 되는가를 파악하기 위하여 다음과 같은 실험 1을 수행하였다. 이 실험에서는 길이가 1024인 620개의 시퀀스들로 구성되는 한국의 KOSPI 주식 데이터를 사용하였다. 또한, 윈도우의 길이는 128을 사용하였으며, 질의 시퀀스의 길이는 256을 사용하였다. 데이터 시퀀스와 질의 시퀀스로부터 각각 디스조인트 윈도우와 슬라이딩 윈도우를 추출하고, 각각을 DFT 변환한 후, 앞쪽의 두 DFT 계수로부터 실수부와 허수부 총 4개의 특징 중 세 개만을 골라 인덱싱을 위한 특징으로 선택하였다.

(그림 1)은 전체 특징 공간상에서 데이터 윈도우 점들과 질의 윈도우 점들이 분포되는 상황을 도면화 한 것이다. 각 사각형은 데이터 윈도우 점을 나타내며, 각 엑스 표시는 질의 윈도우 점을 나타낸다. 말 발굽과 유사한 형태를 가지는 질의 윈도우 점들의 집합은 전체 특징 공간상에서 극히 일부에만 분포됨을 볼 수 있다. 이는 윈도우-조인 처리 시, I_D의 루트 단계에서부터 수많은 MBR들이 이후의 처리 대상에서 제외됨을 나타내는 것이다.

제안된 기법에서 요구하는 추가 작업은 (1) R*-트리의 즉석 구성 및 (2) I_D내의 각 e-MBR을 대상으로 하는 I_Q에 대한 영역 질의 처리이다. 그러나 이 두 작업은 모두 주기억장치 내에서 처리되며, 제4.2절에서 설명한 CPU-최적화를 수행하므로 그 성능 저하 효과는 앞서 언급한 성능 개선 효과들과 비교하여 미미하다.



(그림 1) 특징 공간상에서의 질의 윈도우 점들의 분포

본 연구에서는 대형 시계열 데이터베이스 환경에 대한 성능 평가를 위하여 생성된 랜덤 워크 시퀀스들을 이용한 실험을 수행하였다. 각 데이터 시퀀스는 다음과 같이 생성된다. 첫 요소 값으로서 범위(-10.0, 10.0) 내의 값들 중 하나가 임의로 결정되며, 이후의 요소 값은 각각 직전의 값과 범위(-1.0, 1.0)에서 임의로 선택한 값을 더함으로써 결정된다.

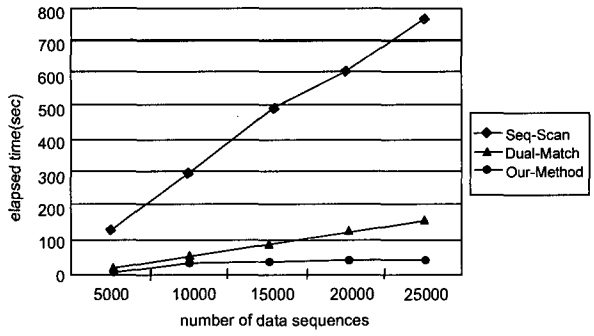
I_D와 I_Q를 위한 R*-트리의 페이지의 크기는 각각 1KB와 64B로 결정하였다. 인덱싱을 위한 특징으로서 앞쪽의 네 DFT 계수로부터 실수부와 허수부 총 8개의 특징 중 여섯 개를 선택하였다. 실험을 위하여 사용된 유사 허용치는 30이며, 성능 지수로서 총 수행 시간을 사용하였다. 또한, 인덱싱을 위한 윈도우의 크기와 질의 시퀀스의 길이를 각각 200과 500으로 설정하였다. 제안된 기법의 비교 대상으로서 제 3장에서 언급한 바와 같이 [10]에서 실질적으로 선택한 인덱스 검색 방법 (2)를 사용하는 Dual-Match를 선택하였다. 또한, 서브시퀀스 매칭을 위한 기본 기법으로서 단순한 순차 스캔 기법도 비교 대상에 포함하였다.

실험에서 사용된 하드웨어 플랫폼은 512MB RAM과 40GB 하드 디스크를 장착한 Pentium IV PCdlau, 소프트웨어 플랫폼은 Windows 2000 Professional 이다. 구현에 사용된 언어는 MS Visual C++이다.

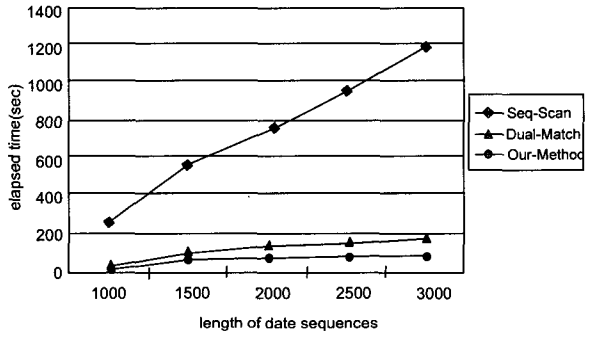
(그림 2)는 데이터 시퀀스 수의 변화에 따르는 실험 결과를 나타낸 것이다. 데이터 시퀀스들의 길이는 1,000으로 설정하였다. X축은 데이터 시퀀스들의 수를 나타내며, Y축은 전체 수행 시간을 나타낸다. 실험 결과에 의하면 제안된 기법은 원래의 Dual-Match의 수행 시간을 약 100%까지 개선시키는 것으로 나타났다. 예상한 바와 같이, 순차 스캔 기법은 원래의 Dual-Match와 비교해서도 크게 떨어지는 성능을 보였다.

(그림 3)은 데이터 시퀀스 길이의 변화에 따르는 성능 결과를 나타낸다. 데이터 시퀀스들의 수는 10,000으로 설정하였다. X축은 데이터 시퀀스들의 수를 나타내며, Y축은 전체 서브시퀀스 매칭의 전체 수행 시간을 나타낸다. (그림 2)에서와 같이, 순차 스캔 기법은 가장 떨어지는 성능을 보였으며, 제

안된 기법은 원래의 Dual-Match와 비교하여 약 100%까지의 상당한 성능 개선을 보였다. (그림 2)와 (그림 3)에 나타난 이러한 성능 결과는 제안된 기법이 디스크 액세스 비용 및 CPU 비용을 성공적으로 최적화하였음을 보이는 것이다.



(그림 2) 데이터 시퀀스 수의 변화에 따르는 성능 결과



(그림 3) 데이터 시퀀스 길이의 변화에 따르는 성능 결과

6. 결 론

본 논문에서는 서브시퀀스 매칭을 효율적으로 처리할 수 있는 새로운 기법을 제안하였다. 본 논문에서는 먼저 서브시퀀스 매칭 시 요구되는 인덱스 검색을 윈도우-조인 문제로 재해석하고, 이 문제에 대한 새로운 해결 방안을 제시하였다. 제안된 기법은 데이터 윈도우를 위한 R*-트리 내에 페이지들을 착오 채택 없이 디스크로부터 단 한번만 액세스하도록 한다. 따라서 제안된 기법은 디스크 액세스 측면에서 최적의 기법이다. 다양한 실험을 통한 성능 평가를 통하여 제안된 기법의 우수성을 규명하였다. 실험 결과에 의하면, 제안된 기법은 전체 수행 시간 측면에서 기존의 Dual-Match의 성능을 약 100%까지 개선시키는 것으로 나타났다.

본 연구에서는 고차원 저주 문제의 해결을 위한 차원 축소 방법으로서 DFT를 사용하였다. 향후 연구로서 이산 코사인 변환(DCT), 웨이블릿 변환[4], Adaptive Piece-wise Constant Approximation(APCA)[8], Segmented Means(SM)[13] 등과 같은 다양한 차원 축소 방법을 이용하는 경우의 제안된 기법의 성능 개선 효과를 정량적으로 검증하는 것을 계획하고 있다.

참 고 문 헌

[1] R. Agrawal, C. Faloutsos and A. Swami, "Efficient Similarity Search in Sequence Databases," In Proc. Int'l. Conf. on Foundations of Data Organization and Algorithms, FODO, pp.69-84, Oct., 1993.

[2] N. Beckmann et al., "The R*-tree : An Efficient and Robust Access Method for Points and Rectangles," In Proc. Int'l. Conf. on Management of Data, ACM SIGMOD, pp.322-331, May, 1990.

[3] T. Brinkhoff, H.-P. Kriegel and B. Seeger, "Efficient Processing of Spatial Joins Using R-Trees," In Proc. ACM Int'l. Conf. on Management of Data, ACM SIGMOD, pp.237-246, 1993.

[4] P. P. Chan and A. W. C. Fu, "Efficient Time-Series Matching By Wavelets," In Proc. IEEE Int'l Conf. on Data Engineering, IEEE ICDE, pp.126-133, 1999.

[5] C. Faloutsos, M. Ranganathan and Y. Manolopoulos, "Fast Subsequence Matching in Time-series Databases," In Proc. Int'l. Conf. on Management of Data, ACM SIGMOD, pp.419-429, May, 1994.

[6] Y.-W. Huang, N. Jing and E. A. Rundensteiner, "Spatial Joins Using R-trees : Breadth-First Traversal with Global Optimizations," In Proc. Int'l. Conf. on Very Large Data Bases, VLDB, pp.396-405, 1997.

[7] S. W. Kim, S. H. Park and W. W. Chu, "An Index-Based Approach for Similarity Search Supporting Time Warping in Large Sequence Databases," In Proc. IEEE Int'l. Conf. on Data Engineering, IEEE ICDE, pp.607-614, 2001.

[8] E. J. Keogh et al., "Locally Adaptive Dimensionality Reduction for Indexing Large Time Series Databases," In Proc. Int'l. Conf. on Management of Data, ACM SIGMOD, pp.419-429, May, 2001.

[9] W. K. Loh, S. W. Kim and K. Y. Whang, "Index Interpolation : An Approach for Subsequence Matching Supporting Normalization Transform in Time-Series Databases," In Proc. ACM Int'l. Conf. on Information and Knowledge Management, ACM CIKM, pp.480-487, 2000.

[10] Y. S. Moon, K. Y. Whang and W. K. Loh, "Duality-Based Subsequence Matching in Time-Series Databases," In Proc. IEEE Int'l Conf. on Data Engineering, IEEE ICDE, pp.263-272, 2001.

[11] D. Rafiei and A. Mendelzon, "Similarity-Based Queries for Time-Series Data," In Proc. Int'l. Conf. on Management of Data, ACM SIGMOD, pp.13-24, 1997.

[12] J. W. Song, K. Y. Whang, Y. K. Lee and S. W. Kim, "Spatial Join Processing Using Corner Transformation," IEEE Trans. on Knowledge and Data Engineering, Vol.11, No.4, pp.688-695, 1999.

[13] B. K. Yi and C. Faloutsos, "Fast Time Sequence Indexing for Arbitrary Lp Norms," In Proc. Int'l. Conf. on Very Large Data Bases, VLDB, pp.385-394, 2000.

김 상 욱

e-mail : wook@hanyang.ac.kr
1989년 서울대학교 컴퓨터공학과(학사)
1991년 한국과학기술원 전산학과(석사)
1994년 한국과학기술원 전산학과(박사)
1991년 미국 Stanford University, Computer Science Department 방문 연구원
1994년~1995년 KAIST 정보전자연구소 전문 연구원
1995년~2000년 강원대학교 컴퓨터정보통신공학부 부교수
1999년~2000년 미국 IBM T. J. Watson Research Center Post-Doc.
2003년~현재 한양대학교 정보통신대학 정보통신학부 부교수
관심분야 : 데이타베이스 시스템, 저장 시스템, 데이타 마이닝, 멀티미디어 정보 검색, 공간 데이터베이스/GIS, 주기억장치 데이타베이스, 트랜잭션 관리

이 헌 길

e-mail : hglee@kangwon.ac.krx
1983년 서울대학교 컴퓨터공학과(학사)
1985년 한국과학기술원 전산학과(석사)
1993년 한국과학기술원 전산학과(박사)
1989년~1993년 (주)삼성전자 선임연구원
1993년~1995년 정보통신부 전산관리소 전산사무관
1995년~현재 강원대학교 공과대학 전기전자정보통신공학부 부교수
관심분야 : 분산처리, 실시간 데이타베이스 시스템, 데이타 마이닝, 컴퓨터 보안, 내장형시스템

박 대 현

e-mail : dhpark@corecom.co.kr
2000년 강원대학교 컴퓨터공학과(학사)
2002년 강원대학교 컴퓨터정보통신공학과(석사)
2002년~2003년 (주)스마트네트테크놀러지 대리
2003년~현재 (주)코아커뮤니케이션즈 주임연구원
관심분야 : 데이타베이스 시스템, 데이타 마이닝, 네트워크