

## 시계열 서브시퀀스 매칭의 윈도우 크기 효과: 정량적 성능 연구

고현길, 정인범  
강원대학교 컴퓨터정보통신공학과

김상욱  
한양대학교 정보통신학부

## Window Size Effect on Time-Series Subsequence Matching: A Qualitative Performance Study

Hyun-Gil Ko and In-Bum Jung  
Department of Computer, Information, and Communications Engineering  
Kangwon National University

Sang-Wook Kim  
College of Information and Communications  
Hanyang University

### 요약

서브시퀀스 매칭은 주어진 질의 시퀀스와 변화의 추세가 유사한 서브시퀀스들을 시계열 데이터베이스로부터 검색하는 연산이다. 본 논문에서는 기존에 제안된 서브시퀀스 매칭 기법인 FRM과 Dual-Match를 대상으로 다양한 실험을 통하여 윈도우 크기 효과를 정량적으로 분석한다. 또한, 이러한 분석 결과를 기반으로 서브시퀀스 매칭 처리의 성능 개선을 위한 향후의 연구 방향을 제시한다.

### 1. 서론

시계열 데이터(time-series data)는 일정한 시간 주기마다 얻어진 실수의 연속적인 값들로 이루어진 데이터이며, 이 데이터들의 집합을 시계열 데이터베이스(time-series database)라 한다[Agr93]. 시계열 데이터베이스에 저장된 데이터를 데이터 시퀀스(data sequence)라 부르며, 질의 시퀀스와 유사한 데이터를 검색하는 연산을 유사 시퀀스 매칭(similar sequence matching)이라고 한다[Agr93][Agr95][Fal94][Moo01]. 유클리드 거리(Euclidean distance)가  $\epsilon$  이하인 두 시퀀스는 서로  $\epsilon$ -매치( $\epsilon$ -match)한다고 한다[Moo01]. 유사 시퀀스 매칭은 데이터 마이닝(data mining) 및 데이터 웨어하우징(data warehousing) 분야에 중요한 연산으로 사용되고 있다 [Che96][Raf97].

기존의 유사 시퀀스 매칭 알고리즘은 크게 전체 매칭(whole matching)과 서브시퀀스 매칭(subsequence matching)으로 구분한다[Agr93][Fal94]. 전체 매칭과는 달리, 서브시퀀스 매칭은 사용되는 데이터 및 질의 시퀀스의 크기에 대한 제약이 없으므로 실제 응용 분야에서 널리 사용된다. 따라서 본 논문은 이러한 서브시퀀스 매칭을 연구의 대상으로 한다.

참고 문헌 [Fal94][Moo01]에서는 서브시퀀스 매칭을 위한 처리 기법을 제안을 하였다. 참고문헌 [Moo01]에서의 제시한 간략한 명칭을 따라 본 논문에서는 참고 문헌 [Fal94]의 기법을 FRM, 참고 문헌 [Moo01]의 기법을 Dual-Match라 부른다. 두 기법 모두 미리 지정된 고정된 크기  $w$ 를 갖는 서브시퀀스인 윈도우(window) 개념을 이용한다. 서브시퀀스 매칭을 위하여 두 기법이 취하는 공통적인 아이디어는 다음과 같다.

먼저, 인덱스 구성을 위하여 각 시퀀스로부터 크기  $w$ 의 윈도우들을 추출하고, 각 윈도우를 DFT 등의 변환 기법을 이

용하여 저차원  $f$ -공간( $f < w$ ) 상의 윈도우 점(window point)으로 변환한다. 효과적인 서브시퀀스 매칭을 위하여 이러한 윈도우 점들을 다차원 인덱스(multidimensional index)[Bec90]에 저장한다.

허용치가  $\epsilon$ 인 서브시퀀스 매칭의 처리를 위하여 크기 1인 질의 시퀀스로부터 크기  $w$ 의 윈도우들을 추출하고, 각 윈도우를 DFT 등의 변환 기법을 이용하여 저차원  $f$ -공간( $f < w$ ) 상의 윈도우 점으로 변환한다. 각 윈도우 점에 대하여  $\epsilon/\sqrt{p}$  ( $p = \lfloor l/w \rfloor$ )를 허용치로 갖는 범위 질의를 다차원 인덱스 상에서 수행한다. 본 논문에서는 이 과정을 인덱스 검색 단계(index search step)라 부른다. 이러한 인덱스 검색 단계의 결과, 질의 시퀀스와  $\epsilon$ -매치 할 가능성이 높은 많은 후보 서브시퀀스(candidate subsequence)들이 반환된다. 그 다음, 착오 채택(false alarm)[Agr93][Fal94]을 해결하기 위하여 이 후보 서브시퀀스들을 포함하는 각 시퀀스를 디스크로부터 익세스하여 후보 서브시퀀스가 질의 시퀀스와 실제로  $\epsilon$ -매치하는가의 여부를 판단한다. 본 논문에서는 이 과정을 후처리 단계(post-processing step)라 부른다.

FRM 및 Dual-Match를 이용한 서브시퀀스 매칭에서는 다차원 인덱스 내에 저장된 윈도우의 크기가 작을수록 착오 채택의 수가 증가하는 경향이 있다[Moo01]. 이러한 경향을 윈도우 크기 효과(window size effect)라 한다. 반면, 저장된 윈도우의 크기가 질의 시퀀스 크기보다 크다면, 해당 다차원 인덱스를 사용할 수 없다. 이 경우에는 순차 검색(sequential scan)에 의하여 서브시퀀스 매칭을 수행해야 하므로 처리 시간이 매우 길어진다. 따라서 다차원 인덱스에 저장될 윈도우의 크기를 올바르게 선택함으로써 전체 서브시퀀스 매칭의 성능을 최적화 할 수 있다.

본 논문에서는 다양한 실험을 통하여 윈도우 크기 효과를

정량적으로 분석한다. 또한, 이 분석 결과를 토대로 향후의 연구 방향을 제시한다.

## 2. 관련 연구

본 장에서는 관련 연구로서 유클리드 거리 기반의 유사 모델을 사용하는 기존의 유사 시퀀스 매칭 기법들을 소개하고, 그 장단점에 관하여 논의한다.

### 2.1. 전체 매칭

참고 문헌 [Agr93]에서는 모든 데이터 시퀀스들과 질의 시퀀스의 크기가 동일하다는 전제 하에 전체 매칭 기법을 제안하였다. 먼저, 크기 1인 각 데이터 시퀀스를 DFT를 이용하여 저차원 f-공간( $f << 1$ ) 상의 점으로 변환하고, 이를 통해 차원 인덱스에 하나인 R\*-트리[Bec90]에 저장함으로써 인덱싱을 수행한다. 여기서, 저차원 변환은 R\*-트리 등의 차차원 인덱스(multidimensional index)가 가지는 고차원 문제(dimensionality curse)[Agr93][Ber96][Web98]를 해결하기 위하여 사용된다.

전체 매칭을 위하여 크기가 1인 질의 시퀀스를 위와 동일한 방법으로 저차원 f-공간상의 한 점으로 변환하고, 변환한 질의 점을 중심점, 허용치  $\epsilon$ 을 범위로 사용하는 범위 질의를 구성한다. 사전에 구성된 R\*-트리에 대하여 이 범위 질의를 처리하는 인덱스 검색 단계(index search step)를 수행한다. 인덱스 검색 단계의 결과, 질의의 점과 유클리드 거리가  $\epsilon$  이하인 모든 데이터 점들과 대응되는 데이터 시퀀스들을 반환하는데, 이를 후보 집합(candidate set)이라 한다.

질의 시퀀스와  $\epsilon$ -매치 하는 데이터 시퀀스들을 후보 집합에 포함시키지 못하는 현상을 착오 기각(false dismissal)이라 한다. 반면, 질의 시퀀스와  $\epsilon$ -매치 하지 않는 일부의 데이터 시퀀스들을 후보 집합에 포함시키는 현상을 착오 채택(false alarm)이라 한다. 참고 문헌 [Agr93]에서는 Parseval 정리를 이용하여 이 기법이 착오 기각을 유발하지 않음을 증명하였다. 그러나 저차원 변환시의 정보 손실로 인한 착오 채택이 발생할 수 있다. 착오 채택을 해결하기 위하여 범위 질의로 구한 후보 집합에 대하여 후처리 단계(post-processing step)를 수행한다. 후처리 단계에서는 디스크로부터 해당 데이터 시퀀스를 액세스하고, 이 시퀀스가 질의 시퀀스와 실제로  $\epsilon$ -매치 하는가의 여부를 조사함으로써 착오 채택을 제거한 최종 질의 결과를 생성한다.

### 2.2. FRM

참고 문헌 [Fal94]에서는 임의의 크기를 갖는 질의 시퀀스와 데이터 시퀀스들을 대상으로 하는 서브시퀀스 매칭 방법 FRM을 제안하였다. FRM은 전체 매칭 기법의 기본 아이디어를 확장한 것으로서, R\*-트리 인덱싱을 위하여 고정된 크기  $w$ 를 갖는 윈도우(window) 개념을 이용한다.

먼저, 크기가  $\text{Len}(S)$ 인 각 데이터 시퀀스  $S$ 로부터 모든 가능한 위치에서 시작되는 크기  $w$ 의 슬라이딩 윈도우(sliding window)들을 추출하고, DFT를 이용하여 각 윈도우를 저차원 f-공간상( $f << w$ )의 한 점으로 변환한다. 본 논문에서는 이 점을 데이터 윈도우 점(data window point)이라 정의한다. 시퀀스  $S$ 로부터 슬라이딩 윈도우들을 추출하므로 각 시퀀스 당 생성되는 데이터 윈도우 점들의 수는  $(\text{Len}(S)-w+1)$ 이 된다. 이 결과, 인덱싱의 대상이 되는 전체 데이터 윈도우 점들의 수가 매우 많아지므로, R\*-트리를 위한 저장 공간이 커지며, 이 결과 인덱스 검색 단계의 성능이 크게 저하된다 [Fal94]. FRM에서는 이러한 문제를 해결하기 위하여 다수의 데이터 윈도우 점들을 포함하는 최소 포함 사각형(minimum bounding rectangle: MBR)들을 구성한 후, 이 MBR들을 R\*-트리[Bec90]에 저장하는 방법을 사용하였다.

서브시퀀스 매칭을 위해서는 크기  $\text{Len}(Q)$ 인 질의 시퀀스  $Q$ 로부터 크기  $w$ 인  $\lfloor \text{Len}(Q)/w \rfloor$  개의 디스조인트 윈도우(disjoint window)들을 추출하고, DFT를 이용하여 각 윈도우를 저차원 f-공간상의 점으로 변환한다. 본 논문에서는 이를 질의 윈도우 점(query window point)이라 정의한다. 각 질의 윈도우 점에 대하여, 그 절의 윈도우 점을 중심점, 허용치  $\epsilon/\sqrt{p}$  ( $p = \lfloor \text{Len}(Q)/w \rfloor$ )를 범위로 사용하는 범위 질의를 구성한다. 사전에 만들어진 R\*-트리에 대하여 이 범위 질의를 처리하는 인덱스 검색 단계를 수행함으로써 후보 집합을 구한다. 후보 집합은 저차원 f-공간상에서 질의 윈도우 점과의 유클리드 거리가  $\epsilon/\sqrt{p}$  이하인 데이터 윈도우 점들을 포함한다. 이러한 점들과 대응되는 데이터 윈도우들을 후보 윈도우(candidate window)라 정의한다. 각 후보 윈도우는 최종 질의 결과에 포함될 가능성이 높은 후보 서브시퀀스(candidate subsequence)와 일대일 대응된다. 최종적으로, 착오 채택을 해결하기 위하여 각 후보 서브시퀀스를 디스크로부터 액세스하여 질의 시퀀스와  $\epsilon$ -매치 하는지를 판단하는 후처리 단계를 수행한다. 참고 문헌 [Fal94]에서는 이러한 서브시퀀스 매칭 기법에서 착오 기각이 발생하지 않음을 증명하였다.

### 2.3. Dual-Match

FRM에서는 인덱싱을 위한 저장 공간의 오버헤드를 줄이기 위하여 개별적인 데이터 윈도우 점들 대신 다수의 윈도우 점들을 포함하는 MBR들을 R\*-트리 내에 저장한다. 그러나 이러한 MBR 내부에는 죽은 공간(dead space)[Bec90]이 존재하게 되므로, 이를 인하여 후보 서브시퀀스의 착오 채택이 발생되며, 이것은 처리 성능의 저하로 직결된다[Moo01]. 참고 문헌 [Moo01]에서는 이러한 문제점을 해결하기 위한 방법으로서 이원성 기반 서브시퀀스 매칭(duality-based subsequence matching: Dual-Match)을 제안하였다.

Dual-Match에서는 크기가  $\text{Len}(S)$ 인 데이터 시퀀스  $S$ 로부터 슬라이딩 윈도우를 추출하고 크기  $\text{Len}(Q)$ 인 질의 시퀀스  $Q$ 로부터 디스조인트 윈도우를 추출하는 FRM과는 반대로 데이터 시퀀스로부터는 디스조인트 윈도우를 추출하고 질의 시퀀스로부터는 슬라이딩 윈도우를 추출하는 방식을 사용한다. 이와 같은 역할 교환을 통하여 Dual-Match는 각 시퀀스로부터  $\lfloor \text{Len}(S)/w \rfloor$  개(여기서  $w$ 는 윈도우의 크기)의 데이터 윈도우 점들만을 추출하게 되므로 R\*-트리에 저장할 데이터 윈도우 점들의 수를 FRM의 약  $1/w$ 로 줄일 수 있다. 이 결과, R\*-트리에 MBR을 저장하는 FRM과는 달리 Dual-Match에서는 윈도우 점 자체를 저장하는 것이 가능해진다. 따라서 MBR 내의 죽은 공간으로 인한 후보 서브시퀀스의 착오 채택이 발생되지 않으므로, 처리 성능이 크게 개선된다. 참고 문헌 [Moo01]에서는 다양한 실험을 통하여 Dual-Match의 검색 성능이 FRM과 비교하여 크게 개선됨을 보였으며, Dual-Match가 착오 기각 없이 서브시퀀스 매칭을 올바르게 수행함을 증명하였다.

### 3. 윈도우 크기 효과

FRM 및 Dual-Match를 이용한 서브시퀀스 매칭에서는 R\*-트리에 저장된 윈도우의 크기가 착을수록 착오 채택의 수가 증가하는 경향이 있다. 예를 들어, 두 R\*-트리  $R1$ 과  $R2$ 에 대하여  $R1$ 이  $R2$ 보다 큰 윈도우를 이용하여 구성되었다고 가정하자. 이때,  $R2$ 를 이용한 인덱스 검색 단계를 통하여 반환된 후보 집합  $S2$  내에는  $R1$ 을 이용한 인덱스 검색 단계를 통하여 반환된 후보 집합  $S1$ 에는 포함되지 않는 후보 서브시퀀스가 존재할 수 있다. 착오 채택이 증가는 후처리 시간의 증가를 초래하며, 전체 수행 시간이 길어지게 된다. 이러한 경향을 윈도우 크기 효과(window size effect)라 한다[Moo01].

라서 큰 원도우를 사용하여 R\*-트리를 구성하는 것은 전체 수행 시간을 개선시키는데 좋은 효과가 있다.

반면, 저장된 원도우의 크기가 질의 시퀀스 크기보다 크다면, 해당 R\*-트리를 사용할 수 없다[Fal94]. 따라서 이 경우에는 순차 검색(sequential scan)에 의하여 서브시퀀스 매칭을 수행해야 하므로 처리 시간이 매우 길어진다. 따라서 다차원 인덱스에 저장될 원도우의 크기를 올바르게 선택함으로써 전체 서브시퀀스 매칭의 성능을 최적화 할 수 있다.

일반적으로 R\*-트리 구성을 위한 원도우 크기는 해당 응용에서 사용 가능한 최소의 질의 시퀀스의 크기  $\min QLen$ (FRM의 경우) 혹은  $\lfloor (\min QLen + 1)/2 \rfloor$  (Dual-Match의 경우)가 된다. 그러나 실제 응용에서 사용되는 질의 시퀀스의 크기는 매우 다양하므로 이와 같이 원도우의 크기와 질의 시퀀스 크기의 차가 큰 경우에는 전체 수행 시간이 매우 커진다.

본 논문에서는 다양한 실험을 통하여 원도우 크기 효과가 전체 서브시퀀스 매칭에 미치는 영향을 정량적으로 분석한다. 또한, 이 분석 결과를 토대로 전체 서브시퀀스 매칭의 처리 성능을 개선시키기 위한 향후의 연구 방향을 제시하고자 한다.

#### 4. 성능 분석

본 장에서는 실험에 의하여 서브시퀀스 매칭의 원도우 크기 효과를 정량적으로 분석한다. 먼저, 제 4.1절에서는 실험 환경을 설명하고, 제 4.2절에서는 실험 결과를 제시하고, 분석 한다.

##### 5.1. 실험 환경

본 연구에서는 실험을 위하여 크기가 1,024인 620개의 시퀀스들로 구성된 한국의 실제 주식 데이터를 사용하였다. 저차원 변환을 위한 특성 추출 함수로 DFT를 사용하였으며, 각 시퀀스로부터 인덱싱을 위하여 6개의 특성을 추출함으로써 6 차원의 R\*-트리를 구성하였다. 첫 번째 DFT 계수에서 항상 0 으로 나타나는 허수 부(imaginary part) 대신, 네 번째 DFT 계수의 실수 부(real part)를 특성으로 사용하였다. 질의 시퀀스는 데이터베이스로부터 선택한 시퀀스로부터 임의의 서브시퀀스를 선택하여 각 요소 값에 적절한 범위<sup>1)</sup> 내의 임의의 값을 더하는 방식으로 변형하여 생성하였다.

성능 평가를 위한 하드웨어 플랫폼은 1.8GHz Pentium III 와 768MB의 주기억장치가 장착된 PC이며, 소프트웨어 플랫폼은 MS Windows 2000 및 Visual Studio.NET이다. 실험 중 다른 프로세스들과의 상호 간섭을 방지하기 위하여 모든 사용자 프로세스들을 제거한 상황에서 실험하였다. 사용된 데이터 및 인덱스 페이지의 크기는 각각 4KB이다. 데이터 및 인덱스 파일을 액세스 할 때, 버퍼링 효과를 제거한 Windows 2000의 I/O 시스템 콜(system call)을 사용함으로써 매 페이지 액세스 시 실제 디스크 액세스를 보장하였다. 또한, 데이터베이스 내의 시퀀스들을 식별자의 순서대로 실제 디스크 상에 연속적으로 저장하였다.

실험의 대상으로서 FRM과 Dual-Match 두 가지 기법을 사용하였다. Dual-Match에서 사용 가능한 최소 원도우 크기는 FRM의 약 1/2이므로 이후부터는 FRM을 기준으로 인자를 설명한다. 원도우 크기 효과를 정량적으로 분석하기 위하여 다음의 두 가지 종류의 실험을 수행하였다. 실험 1의 목적은 크기 64인 원도우들을 대상으로 구성된 R\*-트리를 대상으로 서로 다른 크기 1의 질의 시퀀스 ( $l=64, 128, 256, 512, 1024$ )들에 대한 서브시퀀스 매칭을 수행하였을 때, 질의 시퀀스의 크

기가 변화함에 따라 전체 서브시퀀스 매칭의 성능이 어떻게 변화하는가를 관찰하는 것이다. 실험 2의 목적은 서로 다른 크기  $w$ 의 원도우 ( $w=64, 128, 256, 512, 1024$ )들을 대상으로 구성된 서로 다른 다섯 가지 R\*-트리를 이용하여 크기 1,024인 질의 시퀀스에 대한 서브시퀀스 매칭을 수행하였을 때, 원도우 크기가 변화함에 따라 전체 서브시퀀스 매칭의 성능이 어떻게 변화하는가를 관찰하는 것이다. 허용치  $\epsilon$ 은 각 서브시퀀스 매칭에서 총 20개의 최종 질의 결과가 나오도록 조절함으로써 설정하였다.

성능 지수로는 후보 서브시퀀스 수, 인덱스 검색 단계와 후처리 단계에서 나타나는 각각의 디스크 액세스 수, 인덱스 검색 단계와 후처리 단계에서 요구되는 디스크 액세스 시간과 CPU 처리 시간, 전체 수행 시간을 사용하였다.

##### 5.2. 실험 결과 및 분석

표 1과 표 2는 실험 1에 대한 FRM과 Dual-Match의 수행 결과를 각각 나타낸 것이다.

먼저, 표 1에 나타난 FRM의 결과를 요소별로 살펴보자. 인덱스 검색 단계에서는 질의 시퀀스의 크기가 증가함에 따라 디스크 액세스 수, 디스크 액세스 시간, CPU 처리 시간이 모두 증가하는 것으로 나타났다. 이는 질의 시퀀스와 원도우 크기의 차이가 커짐에 따라 인덱스 검색 단계에서 수행 되어야 할 질의 원도우 수가 증가하므로 인덱스 검색의 횟수가 증가하기 때문이다.

표 1. 질의 시퀀스 크기의 변화에 따른 FRM의 수행 시간 ( $w=64$ ).

질의 크기	후보수	NumDiskAccesses		CPU(sec)		DISK(sec)		총시간
		INDEX	POST	INDEX	POST	INDEX	POST	
64	182,469	39	182,460	0.55	0.38	0.40	10.19	11.52
128	378,826	83	366,107	0.96	1.02	0.91	23.81	26.70
256	863,950	159	780,473	1.86	3.05	1.73	55.32	61.96
512	1,267,357	278	978,487	3.27	7.08	3.02	93.61	106.98
1,024	4,236,189	736	1,991,831	8.64	16.11	8.09	299.01	331.85

표 2. 질의 시퀀스 크기의 변화에 따른 Dual-Match의 수행 시간 ( $w=32$ ).

질의 크기	후보수	NumDiskAccesses		CPU(sec)		DISK(sec)		총시간
		INDEX	POST	INDEX	POST	INDEX	POST	
64	13,676	123	13,221	1.47	0.03	1.19	0.85	3.54
128	39,385	143	36,309	1.75	0.11	1.33	2.32	5.51
256	151,149	182	130,321	2.04	0.53	1.66	9.24	13.47
512	244,923	201	173,923	2.26	1.31	1.73	16.82	22.12
1,024	3,733,568	321	1,465,722	5.53	12.6	3.05	210.21	231.39

또한, 후처리 단계에서도 디스크 액세스 수가 크게 증가하는 것으로 나타났다. 이것은 질의 시퀀스의 크기가 증가함에 따라 후보 서브시퀀스 수가 증가한다는 원도우 크기 효과로 인한 것이다. 표 1의 첫 열에 나타난 후보 서브시퀀스 수는 바로 이러한 원도우 크기 효과를 본 실험에서 발생하였음을 보여주고 있다. 예를 들어, 1,024 크기의 질의 시퀀스를 대상으로 인덱스 검색을 수행한 후 반환되는 후보 서브시퀀스의 수는 원도우의 크기와 동일한 크기 64인 질의 시퀀스를 대상으로 인덱스 검색을 수행한 후 반환되는 후보 서브시퀀스의 수의 20배 이상인 것으로 나타나고 있다. 이 결과, 후처리 단계에서 소요되는 디스크 액세스 시간과 CPU 처리 시간이 모두 크게 증가하는 것이다. 전체 서브시퀀스 매칭의 수행 시간은 크기 64인 질의 시퀀스의 경우 약 11.5초로 나타났으며, 원도우 크기의 16배에 해당하는 크기 1,024인 질의 시퀀스의 경우, 이의 30배에 해당되는 331.8초로 나타났다.

표 2에 나타난 바와 같이 Dual-Match의 경우에도 FRM의 경우와 매우 유사한 경향을 보였다. 그러나 Dual-Match의 경우, FRM에 비해 전체적인 수행 시간이 크게 개선된 것으로

1) 선택된 시퀀스 내에 속하는 요소 값들의 표준 편차를 std라 할 때, 이 범위는  $[\text{std}/10, \text{std}/10]$ 이다.

나타났다. 이는 다수의 윈도우 점들을 포함하는 MBR을 저장하는 FRM과는 달리 윈도우 점들을 직접 저장하는 Dual-Match에서는 차오 채택의 수가 크게 줄어들기 때문이다 [Moo01].

표 3과 4는 실험 2에 대한 FRM과 Dual-Match의 수행 결과를 각각 나타낸 것이다. FRM과 Dual-Match 모두에서 실험 2의 결과는 실험 1과 매우 유사한 경향을 가지는 것으로 나타났다. 즉, 윈도우 크기가 질의 시퀀스의 크기에 가까워 점에 따라 인덱스 검색 단계, 후처리 단계, 전체 서브시퀀스 매칭의 처리에 소요되는 시간은 급격하게 감소함을 볼 수가 있었다. 이는 윈도우 크기가 질의 시퀀스 크기에 가까워 점에 따라 인덱스 검색의 회수와 차오 채택의 수가 모두 크게 줄어듦에 따른 현상이다.

표 3. 윈도우 크기의 변화에 따른 FRM의 수행 시간  
( $i=1,024$ ).

윈도우	후보수	NumDiskAccesses		CPU(sec)		DISK(sec)		총시간
		INDEX	POST	INDEX	POST	INDEX	POST	
64	3,954,138	683	1,901,044	8.23	17.40	6.96	288.15	320.74
128	1,766,568	339	864,294	3.60	8.19	2.96	128.22	142.97
256	707,729	108	337,175	0.86	3.24	0.97	49.5	54.57
512	192,841	35	94,256	0.30	0.90	0.27	14.3	15.77
1,024	66	7	66	0.10	0.01	0.06	0.30	0.47

표 4. 윈도우 크기의 변화에 따른 Dual-Match의 수행 시간  
( $i=1,024$ )

윈도우	후보수	NumDiskAccesses		CPU(sec)		DISK(sec)		총시간
		INDEX	POST	INDEX	POST	INDEX	POST	
32	3,095,704	305	1,306,631	4.79	11.74	2.47	190.02	209.02
64	1,458,322	144	617,216	2.19	5.46	1.19	91.50	100.34
128	647,810	65	281,619	0.95	2.59	0.56	43.11	47.21
256	252,010	28.9	108,112	0.45	1.01	0.31	16.70	18.47
512	106,060	15.8	443,08.8	0.24	0.41	0.17	7.25	8.07

## 5. 결론 및 향후 연구 방향

유사 서브시퀀스 매칭은 시계열 데이터베이스로부터 질의 시퀀스와 유사한 서브시퀀스들을 찾아내는 연산으로서 데이터 마이닝 및 데이터 웨어하우징 분야에 중요한 연산으로 사용되고 있다. 본 논문에서는 기존의 서브시퀀스 매칭 기법인 FRM과 Dual-Match의 성능 개선을 위한 추가 방안에 관하여 연구하였다.

윈도우 크기 효과는 다차원 인덱스 내에 저장된 윈도우의 크기가 질의 시퀀스 크기에 비하여 작을수록 차오 채택의 수가 증가하는 현상이다. 본 논문에서는 다양한 실험을 통하여 윈도우 크기 효과를 정량적으로 분석함으로써 다차원 인덱스 구성을 사용되는 윈도우의 크기가 전체 서브시퀀스 매칭에 얼마나 큰 영향을 미치는가 하는 것을 살펴보았다. 성능 결과에 의하면, 윈도우 크기와 질의 시퀀스 크기의 차가 커질수록 서브시퀀스 매칭의 성능 저하는 매우 심각한 것으로 나타났다.

일반적으로 윈도우의 크기는 응용에서 사용 가능한 최소 절의 시퀀스 크기에 맞추어 결정된다. 또한, 이 응용에서 요구되는 모든 서브시퀀스 매칭 연산들은 이러한 윈도우를 대상으로 구성된 하나의 다차원 인덱스를 이용하여 수행된다. 본 논문에서 제시한 실험 결과를 통하여 우리가 얻을 수 있는 중요한 교훈은 질의 시퀀스의 크기가 다양한 일관적인 응용 환경에서 이러한 접근 방식을 취하는 경우, 윈도우 크기와 차이가 큰 절의 시퀀스를 사용하는 사용자는 매우 불만족스러운 성능을 얻는다는 것이다.

본 저자들은 이러한 문제를 해결하기 위한 향후 연구 방향

으로 다양한 크기의 윈도우들을 대상으로 하는 다수의 다차원 인덱스들을 활용하는 방법을 고려하고 있다. 이 방법은 주어진 질의 시퀀스 보다 작은 윈도우를 중 가장 큰 것에 해당되는 다차원 인덱스를 이용하여 서브시퀀스 매칭을 수행하게 된다. 이 경우, 질의 시퀀스와 윈도우의 크기 차가 작아지므로, 윈도우 크기 효과로 인한 차오 채택의 수가 줄어들게 되며, 이 결과 전체 서브시퀀스 매칭의 성능을 크게 개선시킬 수 있을 것으로 기대한다. 이러한 개념은 참고 문헌 [Loh03]에서 사용된 개념과 유사한 점이 있으므로, 본 연구에서는 이러한 접근 방법을 인덱스 보간법(index interpolation)이라 명명한다.

현재, 본 저자들은 인덱스 보간법과 관련하여 다음의 이슈들을 본 연구의 후속 연구 주제로 추진 중에 있다.

- (1) 주어진 응용에 대하여 몇 개의 다차원 인덱스들을 구성할 것인가 ?
- (2) 각 다차원 인덱스는 어떤 크기의 윈도우를 대상으로 구성할 것인가 ?
- (3) 위의 두 가지 사안을 결정하는데 어떤 기준을 이용할 것인가 ?

이러한 인덱스 보간법에 관한 추가의 연구는 실제 응용 분야에서 서브시퀀스 매칭의 처리 시간을 단축시키는데 큰 도움이 되리라 확신한다.

## [참고 문헌]

- [Agr93] 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.
- [Agr95] R. Agrawal et al., "Fast Similarity Search in the Presence of Noise, Scaling, and Translation in Time-Series Databases," In Proc. Int'l. Conf. on Very Large Data Bases, VLDB, pp. 490-501, Sept. 1995.
- [Bec90] 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.
- [Ber96] S. Berchtold, D. A. Keim, and H.-P. Kriegel, "The X-tree: An Index Structure for High-Dimensional Data," In Proc. Int'l. Conf. on Very Large Data Bases, VLDB, pp. 28-39, 1996.
- [Che96] Chen, M. S., Han, J., and Yu, P. S., "Data Mining: An Overview from Database Perspective," IEEE Trans. on Knowledge and Data Engineering, Vol. 8, No. 6, pp. 866-883, 1996.
- [Fal94] 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.
- [Loh03] Loh Woong-Kee Loh, Sang-Wook Kim, and Kyu-Young Whang, "An Index-Based Subsequence Matching Algorithm Supporting Normalization Transform in Time-Series Databases," Data Mining and Knowledge Discovery Journal, 2003. (accepted to appear)
- [Moo01] Y. S. Moon, K. Y. Whang, and W. K. Loh, "Duality-Based Subsequence Matching in Time-Series Databases," In Proc. Int'l. Conf. on Data Engineering, IEEE ICDE, pp. 263-272, 2001.
- [Raf97] 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.
- [Web98] R. Weber, H.-J. Schek, and S. Blott, "A Quantitative Analysis and Performance Study for Similarity Search Methods in High-Dimensional Spaces," In Proc. Int'l. Conf. on Very Large Data Bases, VLDB, pp. 194-205, 1998.