제22회 한국정보처리학회 추계학술발표대회 논문집 제11권 제2호 (2004. 11)

인덱스 보간법을 이용한 효율적인 시계열 서브시퀀스 매칭

임승환*, 고현길**, 노웅기***, 김상욱*
*한양대학교 정보통신공학과
강원대학교 컴퓨터정보통신공학과 *(주)티맥스데이타 R&D센터

Efficient Time-Series Subsequence Matching Using Index Interpolation

Seung-Hwan Lim*, Hyun-Gil Ko**, Woong-Kee Loh***, Sang-Wook Kim*
*Dept. of Info. & Comm. Eng., Hanyang University
**Dept. of Comp., Info., & Comm. Eng., Kangwon National University ***R&D Center, Tmax Data Co., Ltd.

요 약

서브시퀀스 매칭은 시계열 데이터베이스에서 질의 시퀀스와 유사한 서브시퀀스를 찾아내는 연산이다. 기존의 서브시퀀스 매칭 알고리즘들은 하나의 인덱스만을 사용하여 검색을 수행하기 때문에, 인덱스를 생성하기 위하여 데이터 시퀀스로부터 추출한 윈도우의 크기와 질의 시퀀스의 길이 간의 차이가 커질수록 검색 성능이 급격히 저하되는 문제점을 갖고 있다. 본 논문에서는 이러한 기존 알고리즘의 문제점을 해결하기 위하여 인덱스 보간법에 기반한 새로운 서브시퀀스 매칭 기법을 제안한다. 인덱스 보간법이란 하나 이상의 인덱스를 구축하고 주어진 질의 시퀀스의 길이에 따라 적절한 인덱스를 선택하여 검색을 수행하는 기법이다. 본 논문에서는 서브시퀀스 매칭 비용 공식을 산출하고, 이 비용 공식에 기반하여 제안된 기법의 성능을 최적화 하도록 다수의 인덱스를 구성하는 알고리즘을 제시한다. 마지막으로, 실제 데이터를 이용한 여러 가지 실험을 통하여 제안된 기법의 우수성을 정량적으로 검증한다.

1. 서론

시계열 데이터(time-series data)는 일정한 시간 주기마다 얻어진 연속된 실수 값들로 이루어진 데이터이다. 시계열 데이터로 구성된 데이터베이스를 시계열 데이터베이스(time-series database)라 한다 [1]. 시계열 데이터베이스에 저장된 데이터 시퀀스(data sequence) 중에서 주어진 질의 시퀀스(query sequence)와 유사한 시퀀스를 검색하는 연산을 유사 시퀀스 매칭(similar sequence matching)이라고 한다 [1, 2, 8, 11, 12]. 유사 시퀀스 매칭은 데이터 마이닝(data mining) 분야의 중요한 연산의 하나로 사용되고 있다 [7, 13].

기존의 유사 시퀀스 매칭 기법은 크게 전체 매칭(whole matching)과 서브시퀀스 매칭(subsequence matching)으로 구분한다 [8]. 전체 매칭은 시계열 데이터베이스 내에 저장된 데이터 시퀀스 중에서 입력으로 주어진 질의 시퀀스 Q와 유사한 데이터 시퀀스를 반환한다. 서브시퀀스 매칭은 시계열 데이터베이스 내에 저장된 데이터 시퀀스 중에서 입력으로 주어진 질의 시퀀스 Q와 유사한 서브시퀀스 X를 포함하는 데이터 시퀀스 S와 S 내의 서브시퀀스 X의 위치를 반환한다. 일반적으로, 서브시퀀스 매칭은 전체 매칭에 비하여 다양한 분야에서 응용될 수 있으므로, 본 논문에서는 서브시퀀스 매칭에 연구의 초점을 맞추고자 한다.

기존의 대부분의 유사 시퀀스 매칭 기법에서는 길이가 n인 시퀀스를 n-차원 공간 상의 한 점으로 대응시키며, 두 시퀀스 X, Y 간의 유사성 척도(measure)는 아래의 공식 (1)과 같이 두 n-차원 점들 간의 유클리드 거리(Euclidean distance)로 정의한다 [1, 5, 6, 8, 9, 13, 14]:

$$D(\vec{X},\vec{Y}) = \sqrt{\sum_{1 \le i \le n} (x_i - y_i)^2} \quad (1)$$

여기에서, x_i와 y_i ($1 \le i \le n$)는 각각 두 개의 시퀀스 X와 Y를 구성하는 요소 값들이다. 유사 시퀀스 매칭에서는 아래의 공식 (2)와 같이 서로 간의 거리가 질의 시에 주어진 유사 허용치(tolerance) ε보다 가까운 두 시퀀스 X와 Y를 유사한 시퀀스로 정의하며, 두 시퀀스는 ε-매치(ε-match)한다고 한다 [12].

$$D(\vec{X},\vec{Y}) \le \varepsilon \quad (2)$$

참고문헌 [8, 12]에서는 서브시퀀스 매칭을 위한 기법으로서 FRM과 Dual-Match를 제안하였다. 이러한 기법들은 임의 길이의 데이터 시퀀스와 질의 시퀀스로부터 일정한 크기의 윈도우(window)를 추출하여 서브시퀀스 매칭을 처리한다. 즉, 데이터 시퀀스로부터 윈도우를 추출하여 인덱스를 생성하고, 주어진 질의 시퀀스로부터 윈도우를 추출하여 인덱스를 이용하여 검색을 수행한다.

이러한 윈도우의 크기를 어떻게 결정하는가에 따라 서브시퀀스 매칭의 성능이 많이 좌우된다. 즉, 질의 시퀀스의 길이와 윈도우 크기의 차이가 클수록 검색 성능이 저하된다. 이러한 현상을 윈도우 크기 효과(window size effect)라고 한다[12].

본 논문에서는 이러한 문제점을 해결하고자 인덱스 보간법(index interpolation) [10, 11]에 기반한 새로운 서브시퀀스 매칭 기법을 제안한다. 인덱스 보간법이란 하나 이상의 인덱스를 구축하고 주어진 질의 시퀀스의 길이에 따라 적절한 인덱스를 선택하여 서브시퀀스 매칭을 수행하는 기법이다.

일반적으로, 인덱스 보간법에 기반한 기법에서 많은 수의 인덱스들을 사용할수록 검색 성능은 향상되지만 인덱스 관리 비용도 함께 증가하게 된다. 인덱스 관리 비용은 인덱스를 저장하기 위한 물리적 공간뿐만 아니라, 데이터가 삽입, 삭제, 갱신될 때마다 해당 인덱스를 그에 따라 모두 변경하는 비용을 포함한다. 따라서, 인덱스 보간법에서 최적의 검색 성능을 지원할 수 있는 가능한 한 적은 수의 인덱스들을 선택하여 구성하는 것이 대단히 중요하다.

본 논문의 주요 공헌은 다음과 같다:

(1) 윈도우 크기 효과에 의한 성능 저하를 개선하기 위해 인덱스 보간법에 기반한 새로운 검색 기법을 제안한다.

(2) 물리적 데이터베이스 설계 관점에서 질의 시퀀스의 분포에 따른 검색 비용 공식을 산출한다. 이 검색 비용 공식에 기반하여 제안된 검색 기법의 성능을 최대화할 수 있도록 최적의 인덱스를 구성하는 알고리즘을 제안한다.

(3) 실제 데이터를 이용한 여러 가지 실험을 통하여 제안된 기법으로 인한 성능 개선 효과를 정량적으로 검증한다.

2. 관련 연구

2.1 FRM

FRM [8]은 참고문헌 [1]에서 제시되었던 전체 매칭 기법을 확장하여 제안된 서브시퀀스 매칭 알고리즘이다. FRM은 고정 길이의 윈도우(window)라는 개념을 사용한다. FRM은 인덱싱을 위하여 데이터 시퀀스로부터 일정한 크기 w의 슬라이딩 윈도우(sliding window)들을 추출하고, 서브시퀀스 매칭을 위하여 질의 시퀀스로부터 동일한 크기 w의 디스조인트 윈도우 (disjoint window)들을 추출하는 방법을 사용한다.

FRM은 인덱싱 단계에서 데이터베이스 내의 모든 데이터 시퀀스 S로부터 길이 w인 슬라이딩 윈도우를 추출하고, 이산 푸리에 변환(Discrete Fourier Transform, DFT)을 이용하여 각 슬라이딩 윈도우를 f ($<< w$) 차원 공간 상의 한 점으로 변환한다. 이러한 점을 데이타 윈도우 점이라 부른다. 길이 $Len(S)$인 데이터 시퀀스로부터 추출 가능한 슬라이딩 윈도우의 개수는 $(Len(S) - w + 1)$이다. 효과적인 서브시퀀스 매칭을 위하여 이러한 윈도우 점들을 인덱싱하기 위한 자료 구조로서 다차원 인덱스(multidimensional index)의 하나인 R*-트리 [3]를 사용한다.

FRM의 서브시퀀스 매칭 단계는 크게 인덱스 검색 단계와 후처리 단계로 구성된다. 인덱스 검색 단계에서는 주어진 질의 시퀀스 Q를 길이 w인 디스조인트 윈도우로 분할하여 각각을 인덱싱 단계에서와 같은 방법으로 f 차원 상의 한 점으로 변환한다. 이러한 점을 질의 윈도우 점이라 부른다. 각 질의 윈도우 점에 대하여 질의 시에 주어진 허용치 ε으로부터 구해진 새로운 허용치 ε' 범위 내에 존재하는 모든 데이터 윈도우 점들을 R*-트리를 통하여 검색한다. 이때, 새로운 허용치 ε'은 아래의 공식 (3)을 이용하여 구하며, 여기에서 p는 질의 시퀀스로부터 추출한 디스조인트 윈도우의 개수이다.

$$\varepsilon' = \frac{\varepsilon}{\sqrt{p}} \left(p = \left\lfloor \frac{Len(Q)}{w} \right\rfloor \right) \quad (3)$$

인덱스 검색 단계에서 반환된 데이터 윈도우를 포함하는 모든 서브시퀀스들로 후보 집합을 구성한다. 후보 집합 내에는 최종 질의 결과로 반환되지 않는 서브시퀀스들이 포함되어 있으며, 이들을 착오 채택(false alarm)이라고 부른다. 이러한 착오 채택을 제거하기 위하여 후처리 단계를 거치게 된다. 후처리 단계에서는 후보 집합 내에 존재하는 각 후보 서브시퀀스에 대하여 이를 포함하는 실제 데이터 시퀀스를 디스크로부터 읽어 질의 시퀀스와의 유클리드 거리가 허용치 ε 이하인지의 여부를 검토함으로써 착오 채택을 제거한다. FRM에서는 주어진 응용에서 발생할 수 있는 최소의 질의 시퀀스 길이 $\min(Len(Q))$를 윈도우의 크기 w로 사용한다.

FRM에서는 하나의 데이터 시퀀스 S로부터 $(Len(S) - w + 1)$ 개의 슬라이딩 윈도우를 추출하므로, 전체 데이터베이스에서 보면 대단히 많은 개수의 f 차원 윈도우 점이 생기게 된다. 이러한 점들을 모두 인덱스에 저장하기 위해서는 큰 저장 공간이 필요하게 되며 동시에 인덱스 검색 성능도 많이 떨어진다. 참고문헌 [8]에서는 이러한 문제를 해결하기 위해 윈도우 점들을 개별적으로 인덱스에 저장하지 않고, 다수의 윈도우 점들을 포함하는 하나의 최소 포함 사각형(minimum bounding rectangle, MBR)을 구성한 후 이 MBR들을 인덱스에 저장하는 방법을 사용한다. 데이터 윈도우 점들을 직접 인덱스에 저장하지 않고 대신 MBR을 구성하여 인덱스에 저장하는 방법은 저장 공간을 줄일 수 있으나, 검색 결과 착오 채택을 크게 증가시킬 수 있다는 단점을 갖고 있다 [12].

2.2 Dual-Match

Dual-Match는 앞에서 설명한 FRM의 단점을 개선하고자 참고문헌 [12]에서 제안한 서브시퀀스 매칭 기법이다. Dual-Match에서는 FRM과 반대로 데이터 시퀀스로부터 디스조인트 윈도우들을 추출하고, 질의 시퀀스로부터 슬라이딩 윈도우들을 추출하는 방법을 사용한다. 즉, 길이 $Len(S)$인 하나의 데이터 시퀀스 S로부터 추출되는 길이 w의 윈도우는 모두 $\text{FLOOR}(Len(S) / w)$ 개이며, 길이 $Len(Q)$인 질의 시퀀스 Q로부터 추출되는 윈도우는 모두 $(Len(Q) - w + 1)$ 개이다.

Dual-Match에서 데이터 시퀀스로부터 디스조인트 윈도우들을 추출함으로 인하여 다차원 인덱스에 저장해야 하는 데이터 윈도우 점들의 개수가 FRM에 비하여 약 $(1 / w)$로 줄어들게 된다. 따라서, 다수의 데이터 윈도우 점을 하나의 MBR로 구성하여 인덱스에 저장하는 FRM과 달리 Dual-Match에서는 데이터 윈도우 점들을 직접 인덱스에 저장한다. 이러한 인덱스 구성 방법에 의해 Dual-Match에서는 착오 채택을 크게 줄일 수 있다. 서브시퀀스 매칭에서는 착오 채택의 수가 전체 검색 성능에 큰 영향을 미치므로 [5], Dual-Match는 FRM에 비하여 크게 성능을 향상시킬 수 있다. Dual-Match에서는 서브시퀀스 매칭 응용에서 주어질 수 있는 최소의 질의 시퀀스 길이를 $\min(Len(Q))$라 할 때, $\text{FLOOR}((\min(Len(Q)) + 1) / 2)$를 윈도우의 크기 w로 사용한다.

3. 제안하는 기법

본 장에서는 서브시퀀스 매칭 수행 시에 윈도우 크기 효과에 의한 착오 채택을 줄이기 위하여 인덱스 보간법(index interpolation) [10, 11]을 이용하는 새로운 기법을 제안한다. 인덱스 보간법은 서로 다른 윈도우 크기를 이용하는 다수의 인덱스를 구축하여 질의 시퀀스의 길이에 따라 가장 적절한 하나의 인덱스를 선택하여 검색을 수행하는 기법이다.

인덱스 보간법에 기반한 검색 기법에서는 일반적으로 많은 인덱스들을 구성할수록 서브시퀀스 매칭 성능은 향상되지만, 이와 함께 인덱스 관리 비용도 증가하게 된다. 인덱스 관리 비용은 인덱스를 저장하기 위한 물리적 공간뿐만 아니라, 데이터가 삽입, 삭제, 갱신될 때마다 해당 인덱스들을 변경해야 하는 비용을 포함한다. 따라서, 인덱스 보간법에서는 최적의 검색 성능을 지원할 수 있는 가능한 한 적은 수의 인덱스들을 선택하여 구성하는 것이 대단히 중요하다. 본 장에서는 물리적 데이터베이스 설계 관점에서 효율적인 서브시퀀스 매칭을 지원하기 위하여 최적의 윈도우 크기들을 선택하여 인덱스를 구축하는 방안에 대하여 논의한다.

3.1 기본 아이디어

기존의 서브시퀀스 매칭 알고리즘에서 윈도우의 크기는 응용에서 입력 가능한 최소 질의 시퀀스의 길이(FRM의 경우) 혹은 최소 질의 시퀀스 길이의 절반(Dual-Match의 경우)으로 결정된다 [8, 12]. 기존의 서브시퀀스 매칭 알고리즘은 이렇게 선택된 크기의 윈도우들을 대상으로 단 하나의 인덱스만을 구축하여 검색에 이용한다. 이러한 방식은 질의 시퀀스의 길이가 매우 다양한 일반적인 응용 환경에서 윈도우 크기와 차이가 큰 질의 시퀀스에 대하여 매우 불만족스러운 성능을 나타낸다. 따라서, 이렇게 다양한 길이의 질의 시퀀스에 대한 서브시퀀스 매칭에서 만족할만한 성능을 보장하기 위해서는 윈도우 크기 효과를 고려하여 다양한 크기의 윈도우들에 대한 다수의 인덱스들을 구축하여 사용하는 인덱스 보간 전략이 주효할 것이다.

인덱스 보간법에 기반한 서브시퀀스 매칭을 위하여 구축된 인덱스를 w-인덱스(w-index)라고 부른다 [10, 11]. 여기에서 w는 해당 인덱스를 구축한 윈도우의 크기를 나타낸다. 서브시퀀스 매칭은 하나의 w-인덱스를 선택하여 처리하게 되는데, 주어진 질의 시퀀스에 대한 최적의 w-인덱스를 선택하기 위한 공식은 아래와 같다:

$$w_{\max} = \max\{w_i \mid w_i \le Len(Q)\ (1 \le i \le k)\} \text{ (FRM) 또는}$$

$$w_{\max} = \max\{w_i \mid w_i \le \lfloor (Len(Q)+1)/2 \rfloor\ (1 \le i \le k)\} \text{ (Dual-Match)} \quad (4)$$

여기에서, k는 w-인덱스의 개수이며, w_{max} 값은 FRM과 Dual-Match에 따라 다른 값을 갖는다. 공식 (5)에 의하여 w_{max}가 구해지면 w_{max} 크기의 윈도우들을 대상으로 구축된 w-인덱스를 이용하여 서브시퀀스 매칭을 수행한다. 어떤 윈도우 크기 w를 대상으로 w-인덱스를 생성할 것인가에 대해서는 다음 절에서 상세히 설명한다.

아래의 보조 정리 1은 인덱스 보간법을 이용한 서브시퀀스 매칭 기법의 견고성(robustness)을 보이기 위한 것이다.

보조 정리 **1.** 공식 (4)에 의해 선택된 w-인덱스를 이용하여 수행한 서브시퀀스 매칭에서는 착오 기각이 발생하지 않는다.

증명: 생략

3.2 다수의 인덱스 구성 방안

인덱스 보간법에 기반하여 효율적인 서브시퀀스 매칭의 수행을 위한 w-인덱스의 구축을 위해서는 아래와 같은 세 가지 주요 사항들을 고려하여야 한다:

(1) 주어진 응용에 대하여 몇 개의 w-인덱스들을 구성할 것인가?

(2) 각 w-인덱스는 어떤 크기의 윈도우들을 대상으로 구성할 것인가?

(3) 위의 두 가지 사항을 결정하는 데에 어떤 기준을 적용할 것인가?

인덱스 보간법은 하나 이상의 인덱스를 사전에 구성해 두고, 주어진 질의 시퀀스에 대하여 적절한 인덱스를 선택하여 서브시퀀스 매칭을 수행한다는 기본적인 원칙일 뿐 인덱스들을 어떻게 구성할 것인가는 경우에 따라 다르다 [10, 11]. 본 논문에서는 물리적 데이터베이스 설계 기법을 이용한 최적의 다수의 인덱스 구성 방안에 대하여 논의한다. 대부분의 응용에서 사용되는 질의들의 경향이 과거에 사용되었던 경향과 유사하다는 가정 하에, 제안된 기법은 과거의 질의 사용 정보를 바탕으로 질의 시퀀스들의 길이 분포와 생성할 w-인덱스의 개수 k를 입력으로 받아 최적의 w-인덱스들을 구성할 윈도우 크기 $w_1, w_2, \ldots, w_k$를 결정한다. 이러한 물리적 데이타베이스 설계를 이용하는 기법은 비록 같은 개수의 인덱스를 사용하더라도, 일정 간격의 윈도우들을 대상으로 구성된 w-인덱스들을 이용하는 경우와 비교하여 더 좋은 성능을 얻을 수 있다.

본 절에서는 먼저 전체 서브시퀀스 매칭 처리를 위한 비용 공식을 도출하고, 그 공식에 기반하여 전체 질의 시퀀스들을 대상으로 최적의 서브시퀀스 매칭 수행을 위한 윈도우 크기들을 선정하는 알고리즘을 제시한다. 검색 비용 공식을 도출하기 위하여 전체 질의 시퀀스를 같은 길이를 갖는 질의 시퀀스의 그룹 $Q_1, Q_2, \ldots, Q_g$ ($Len(Q_1) < Len(Q_2) < \ldots < Len(Q_g)$)로 분할한다. 각 그룹에 대하여 공식 (4)에 의하여 선택된 윈도우의 크기를 $w_{max}(Q_1), w_{max}(Q_2), \ldots, w_{max}(Q_g)$ ($w_{max}(Q_1)$ $w_{max}(Q_2)$ ... $w_{max}(Q_g)$)라 한다. 여기에서, 인덱스의 개수 k는 질의 시퀀스 그룹의 개수 g보다 작거나 같다고 가정한다.

이러한 설정 하에 인덱스 보간법에 기반한 전체 질의 시퀀스들을 대상으로 서브시퀀스 매칭을 수행하기 위한 전체 검색 비용 T를 구하면 아래의 공식과 같다:

$$T = \sum_{1 \le i \le g} \left(\frac{Len(Q_i)}{w_{\max}(Q_i)} \right) \cdot F_i \qquad (5)$$

여기에서, F_i (1 i g)는 각 질의 시퀀스 그룹의 사용 빈도이며, Q_i 그룹 내의 질의 시퀀스들의 개수를 전체 질의 시퀀스들의 개수로 나눈 값이다.

공식 (5)의 비용 T를 최소로 하는 윈도우 크기 $w_{max}(Q_i)$ (1 i k)를 구함에 있어서 모든 윈도우 조합에 대하여 검토하기 위한 알고리즘의 실행 시간 복잡도(time complexity)는 $O(M^k)$이 되며, 여기에서 M은 질의 시퀀스의 최대 길이이고 k는 w-인덱스의 개수이다. 그러한 알고리즘은 지수적 실행 시간 복잡도를 가지므로 M 또는 k 값이 증가함에 따라 알고리즘의 계산 시간이 기하급수적으로 증가하게 된다.

따라서, 본 논문에서는 이러한 문제를 해결하기 위하여 윈도우 크기 선정을 위한 휴리스틱(heuristic) 알고리즘을 제시한다. 그림 1에 보인 알고리즘은 주어진 질의 시퀀스들의 길이 분포와 w-인덱스의 개수 k를 입력으로 받아, 공식 (5)에 보인 비용 T 값을 최소화함으로써 전체 서브시퀀스 매칭의 성능을 최대화하기 위한 윈도우 크기 $w_{max}(Q_i)$ (1 i k)를 반환한다. 이 알고리즘은 두 개의 중첩된 for 루프만을 포함하고 있으므로, 실행 시간 복잡도는 $O(M \cdot k)$이다.

```
Procedure GetWindowSizes
(1)   Compute w_max(Q1);
(2)   for i = 2 .. k do
(3)       for each possible w other than w_max(Qi) (1  j < i) do
(4)           Compute T;                          // using Eq. (5)
(5)           If T is minimum then
(6)               w_max(Qi) = w;
(7)           end if
(8)       end for
(9)   end for
end.
```

그림 1. 윈도우 크기 선정 알고리즘.

그림 1의 알고리즘을 라인 별로 설명하면 다음과 같다. 라인 (1)에서는 $w_{max}(Q_1)$을 구한다. 윈도우 크기는 질의 시퀀스의 길이보다 작거나 같아야 하므로, FRM과 Dual-Match에 대하여 $w_{max}(Q_1)$은 각각 $Len(Q_1)$ 또는 FLOOR(($Len(Q_1)$ + 1) / 2)이 되어야 한다. 만약 $w_{max}(Q_1)$이 이러한 크기보다 작다면 윈도우 크기 효과에 의하여 착오 채택이 발생하여 성능이 떨어지고, 만약 $w_{max}(Q_1)$이 이보다 크다면 서브시퀀스 매칭에 w-인덱스를 이용할 수 없게 된다.

라인 (2) ~ (9)는 $w_{max}(Q_i)$ (2 i k)를 구하기 위한 부분이며, 라인 (3) ~ (8)은 하나씩의 $w_{max}(Q_i)$를 구하는 부분이다. 라인 (3)에서 $w_{max}(Q_i)$를 구함에 있어서 중복이 발생하지 않도록 이전까지 구해진 구해진 $w_{max}(Q_j)$ (1 $j < i$)를 제외한 나머지 윈도우 크기 w를 검토 대상으로 한다. 라인 (4)에서는 이전에 정해진 $w_{max}(Q_j)$ (1 $j < i$)와 윈도우 크기 w를 공식 (5)에 대입하여 T 값을 산출한다. 라인 (5) ~ (7)에서는 만약 라인 (4)에서 구해진 T 값이 현재까지 얻어진 최소 비용 값보다 작다면 현재 윈도우 크기 w를 $w_{max}(Q_i)$로 할당한다.

4. 성능 평가

4.1 실험 환경

본 실험에서는 성능 평가를 위하여 길이가 1024인 620 개의 데이터 시퀀스들로 구성된 한국의 실제 주식 데이터를 이용하였다. 질의 시퀀스의 길이는 64 ~ 1024 사이에서 32 간격의 길이를 가지며, 같은 길이를 갖는 질의 시퀀스를 같은 그룹으로 구성하였다 (총 31 그룹). 본 실험에서 동일한 수의 질의 시퀀스들을 포함하는 그룹의 개수와 그 그룹에 포함된 질의 시퀀스의 개수는 실제 응용과 유사하도록 표 1과 같이 비중을 달리하여 조정하였으며, 총 216 개의 질의를 수행하는 데에 걸린 실행 시간의 합을 평가지수로 삼았다. 각 질의 시퀀스에 대한 허용치 ε은 서브시퀀스 매칭의 결과로 20 개의 서브시퀀스를 반환하도록 조정하였다.

실험을 위한 하드웨어 플랫폼은 1.8 GHz Pentium 4 프로세서와 512 MB의 메모리를 장착한 PC를 사용하였으며 운영체제는 MS Windows 2000을 이용하였다. 본 실험에서는 일정한 실험 결과를 얻기 위하여 운영 체제에서 제공하는 버퍼링(buffering) 기능을 사용하지 않고 바로 디스크를 액세스하는 시스템 I/O 함수를 이용하였다. 윈도우를 저장하기 위한 다차원 인덱스로는 R*-트리 [3]를 사용하였다. 인덱싱을 위한 저차원 변환은 DFT를 통하여 수행하였으며, 특성 추출 계수 f = 6으로 하였다.

질의 시퀀스 그룹의 개수	그룹 내 질의 시퀀스의 개수	질의 시퀀스 그룹의 개수	그룹 내 질의 시퀀스의 개수
4	30	6	5
5	10	16	1

표 1. 그룹 내에 포함된 질의 시퀀스의 개수.

본 실험에서의 비교 대상은 세 가지이다: (A) 단일 인덱스만을 이용

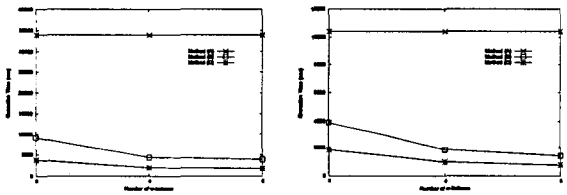
하는 기존의 FRM 및 Dual-Match, (B) FRM 및 Dual-Match를 균일 간격의 *w*-인덱스를 이용하도록 확장한 기법, (C) FRM 및 Dual-Match를 본 논문에서 제안한 기법으로 생성한 *w*-인덱스를 이용하도록 확장한 기법. 본 논문에서는 이들을 각각 방법 (A), (B), (C)라고 부른다.

4.2 실험 결과

실험은 FRM과 Dual-Match에 대하여 각각 수행하였다. FRM은 방법 (A)에서는 $w = 64$인 하나의 인덱스만을 사용하였으며, 방법 (B)에서는 $w = 64, 304, 544, 784, 1024$인 5개의 *w*-인덱스를 사용하였고, 방법 (C)에서는 $w = 64, 224, 384, 768, 896$인 5개의 *w*-인덱스를 사용하였다. Dual-Match은 방법 (A)에서는 $w = 32$인 하나의 인덱스만을 사용하였으며, 방법 (B)에서는 $w = 32, 152, 272, 392, 512$인 5개의 *w*-인덱스를 사용하였고, 방법 (C)에서는 $w = 32, 112, 192, 384, 448$인 5개의 *w*-인덱스를 사용하였다.

그림 2는 실험의 결과를 보인 것이다. 그림 2(a)는 FRM에 대하여, 그림 2(b)는 Dual-Match에 대하여 실험한 결과이다. 그림 2에서 가로 축은 *w*-인덱스의 개수, 세로 축은 실행 시간을 초(second) 단위로 나타낸 것이다.

두 가지 기법 모두에 대하여 기존의 방식을 그대로 적용한 방법 (A)에 비하여 여러 개의 *w*-인덱스를 이용한 방법 (B)가 높은 성능을 보였고, 일정 간격의 *w*-인덱스를 이용한 방법 (B)에 비하여 제안된 기법에 의하여 정해진 *w*-인덱스를 이용한 방법 (C)가 더 좋은 성능을 보였다. 그림 2(a)에서 5 개의 *w*-인덱스를 이용한 경우 방법 (C)는 방법 (A)에 비하여 약 18.4 배, 방법 (B)에 비하여 약 2.1 배 성능이 향상되었다. 그림 2(b)에서 5 개의 *w*-인덱스를 이용한 경우 방법 (C)는 방법 (A)에 비하여 약 13.3 배, 방법 (B)에 비하여 약 1.8 배 성능이 향상되었다.



(a) FRM. (b) Dual-Match.

그림 2. *w*-인덱스의 개수에 의한 성능 비교.

5. 결론

본 논문에서는 기존의 기법에서 윈도우 크기 효과에 의하여 검색 성능이 저하되는 문제점을 해결하고자 인덱스 보간법 [10, 11]에 기반한 새로운 서브시퀀스 매칭 기법을 제안하였다. 인덱스 보간법이란 하나 이상의 *w*-인덱스를 구축하고, 주어진 질의 시퀀스의 길이에 따라 하나의 적절한 *w*-인덱스를 선택하여 검색을 수행함으로써 서브시퀀스 매칭 성능을 향상시키는 기법이다.

본 논문의 주요 공헌은 다음과 같다. 물리적 데이터베이스 설계 관점에서 서브시퀀스 매칭 처리를 위한 검색 비용 공식을 산출하였고, 윈도우 크기 효과에 의한 성능 저하를 개선하기 위해 인덱스 보간법에 기반한 새로운 검색 기법을 제안하였으며, 검색 비용 공식에 기반하여 제안된 검색 기법의 성능을 최적화할 수 있도록 인덱스를 구성하는 알고리즘을 제시하였다. 마지막으로, 실제 데이터를 이용한 여러 가지 실험을 통하여 제안된 기법이 기존의 단순한 검색 기법에 비하여 성능이 크게 개선됨을 정량적으로 검증하였다.

실험 결과에 의하면 FRM의 경우, 제안한 기법이 단일 인덱스를 사용한 경우와 비교하여 약 18.4 배의 성능향상을 보였으며, 일정 구간의 인덱스를 사용한 경우와 비교하여 약 2.1 배의 성능 향상을 보였다. 또한, Dual-Match의 경우, 단일 인덱스를 사용한 경우와 비교하여 약 13.3 배, 일정 구간의 인덱스를 사용한 경우와 비교하여 약 1.8 배의 성능 향상을 보였다.

6. 감사의 글

이 논문은 2003년도 한국학술진흥재단의 선도과학자 연구비 지원 (KRF-2003-041-D00486)을 통하여 수행되었습니다.

참고문헌

[1] R. Agrawal et al., "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] R. Agrawal et al. "Fast Similarity Search in the Presence of Noise, Scaling, and Translation in Time-Series Database," In Proc. Int'l Conf. on Very Large Data Bases,VLDB, pp. 490-501, Sept. 1995.

[3] 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.

[4] C. Chatfield, The Analysis of Time-Series: An Introduction, 3rd Ed., Chapman and Hall, 1984.

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

[6] K. K. W. Chu and M. H. Wong, "Fast Time-Series Searching with Scaling and Shifting," In Proc. Int'l Symposium on Principles of Database Systems, ACM PODS, pp. 237-248, May 1999.

[7] M. S. Chen et al., "Data Mining: An Overview from Database Perspective," IEEE Trans. on Knowledge and Data Engineering, Vol. 8, No. 6, pp. 866-883, June 1996.

[8] C. Faloutsos et al., "Fast Subsequence Matching in Time-series Databases," In Proc. Int'l Conf. on Management of Data, ACM SIGMOD, pp. 419-429, May 1994.

[9] D. Q. Goldin and P. C. Kanellakis, "On Similarity Queries for Time-Series Data: Constraint Specification and Implementation," In Proc. Int'l Conf. on Principles and Practice of Constraint Programming, pp. 137-153, Sept. 1995.

[10] W. K. Loh et al., "Index Interpolation: A Subsequence Matching Algorithm Supporting Moving Average Transform of Arbitrary Order in Time-Series Databases," IEICE Transactions on Information and Systems, Vol. E84-D, No. 1, pp.76-86, Jan. 2001.

[11] W. K. Loh et al., "A Subsequence Matching Algorithm that Supports Normalization Transform in Time-Series Databases," Data Mining and Knowledge Discovery, Vol. 9, No. 1, pp. 5-28, July 2004.

[12] Y. S. Moon et al., "Duality-Based Subsequence Matching in Time-Series Databases," In Proc. Int'l Conf. on Data Engineering, IEEE ICDE, pp. 263-272, Apr. 2001.

[13] 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, June 1997.

[14] D. Rafiei, "On Similarity-Based Queries for Time Series Data," In Proc. Int'l Conf. on Data Engineering, IEEE ICDE, pp. 410-417, Mar. 1999.

[15] R. Weber et al., "A Quantitative Analysis and Performance Study for Similarity Search Methods on High-Dimensional Spaces," In Proc. Int'l Conf. on Very Large Data Bases, VLDB, pp. 194-205, Aug. 1998.