인덱스 보간법에 기반한 효율적인 서브시퀀스 매칭 기법

노 웅 기[†] · 김 상 욱[††]

요 약

서브시퀀스 매칭은 데이터 마이닝 분야에서 중요한 연산 중의 하나이다. 기존의 서브시퀀스 매칭 알고리즘들은 하나의 인덱스만을 사용하여 검색을 수행하며, 인덱스를 생성하기 위하여 데이터 시퀀스로부터 추출한 윈도우의 크기와 질의 시퀀스의 길이 간의 차이가 커질수록 검색 성능이 급격히 저하된다. 본 논문에서는 이러한 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 먼저 사전 실험을 통하여 서브시퀀스 매칭을 수행하는 데에 있어 질의 시퀀스 길이와 윈도우 크기 간의 차이로 인한 성능의 변화를 관찰하고, 이 관찰을 통하여 물리적 데이터베이스 설계 관점에서 질의 시퀀스의 길이 분포에 따른 검색 비용 공식을 산출한다. 다음에, 윈도우 크기 효과에 의한 성능 저하를 개선하기 위해 인덱스 보간법에 기반한 새로운 검색 기법을 제안한다. 또한, 검색 비용 공식에 기반하여 제안된 검색 기법의 성능을 최적화할 수 있도록 다수의 인덱스를 구성하는 알고리즘을 제시한다. 마지막으로, 실제 데이터와 합성 데이터를 이용한 여러 가지 실험을 통하여 제안된 기법의 우수성을 검증한다.

키워드 : 서브시퀀스 매칭, 인덱스 보간법, 윈도우 크기 효과, 물리적 데이터베이스 설계, 검색 비용 공식

An Efficient Subsequence Matching Method Based on Index Interpolation

Woong-Kee Loh[†] · Sang-Wook Kim[††]

ABSTRACT

Subsequence matching is one of the most important operations in the field of data mining. The existing subsequence matching algorithms use only one index, and their performance gets worse as the difference between the length of a query sequence and the size of windows, which are subsequences of a same length extracted from data sequences to construct the index, increases. In this paper, we propose a new subsequence matching method based on index interpolation to overcome such a problem. An index interpolation method constructs two or more indexes, and performs search ing by selecting the most appropriate index among them according to the given query sequence length.

In this paper, we first examine the performance trend with the difference between the query sequence length and the window size through preliminary experiments, and formulate a search cost model that reflects the distribution of query sequence lengths in the view point of the physical database design. Next, we propose a new subsequence matching method based on the index interpolation to improve search performance. We also present an algorithm based on the search cost formula mentioned above to construct optimal indexes to get better search performace. Finally, we verify the superiority of the proposed method through a series of experiments using real and synthesized data sets.

Key Words : Subsequence Matching, Index Interpolation, Window Size Effect, Physical Database Design, Search Cost Formula

1. 서 론

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 데이터의 대표적인 예는 주가 및 환율 데이터, 기온 데이터, 물품 판매 데이터, 의료 측정 데이터 등이 있다[2, 4, 8]. 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]. 시계열 데이터 간의 유사성 문제의 예로는 주가가 유사하게 변동하는 주식 종목의 검색, 최근 기간과 유사한 기온 패턴을 갖는 과거 기간의 검색, 과거의 환율 변동 경향을 바탕으로 한 현재 환율 변동 예측 등이 있다[2, 4, 8].

기존의 유사 시퀀스 매칭 기법은 크게 전체 매칭(whole matching)과 서브시퀀스 매칭(subsequence matching)으로

※ 이 논문은 2003년도 한국학술진흥재단의 선도과학자 연구비 지원(KRF-2003-041-D00486)과 제주대학교 IT연구센터(텔리매틱스 요소 기술 연구)를 통한 연구비 지원을 받았습니다.
† 정 회 원 : 한국과학기술원 전산학과 초빙교수
†† 종신회원 : 한양대학교 정보통신대학 정보통신학부 부교수
논문접수 : 2004년 10월 4일, 심사완료 : 2005년 3월 24일

구분한다[8]. 전체 매칭은 시계열 데이터베이스 내에 저장된 데이터 시퀀스 중에서 입력으로 주어진 질의 시퀀스 Q와 유사한 데이터 시퀀스를 반환한다. 이때, 데이터베이스 내의 모든 데이터 시퀀스와 질의 시퀀스 Q의 길이는 모두 동일하다. 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. 이때, 데이터베이스 내의 데이터 시퀀스와 질의 시퀀스 Q는 서로 다른 임의의 길이를 갖는 것이 가능하다. 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 기법들을 제안하였다. 본 논문에서는 참고문헌 [12]에서와 같이 각각의 기법을 FRM과 Dual-Match라 부르며, 제2장에서 좀더 상세히 설명한다. 이러한 서브시퀀스 매칭 기법들은 임의 길이의 데이터 시퀀스와 질의 시퀀스로부터 일정한 크기의 윈도우(window)를 추출하여 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 하며[12], 제3장에서 좀더 상세히 설명한다. FRM과 Dual-Match에서는 최소 질의 시퀀스 길이(FRM의 경우) 혹은 최소 질의 시퀀스 길이의 절반(Dual-Match의 경우) 이하의 크기를 갖는 윈도우를 이용하여야 한다.

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 하나의 적절한 인덱스를 선택하여 서브시퀀스 매칭을 수행하는 기법이다. 인덱스 보간법은 실제 응용에 따라 인덱스 구축 방법, 인덱스 선택 방법, 세부적인 처리 알고리즘 등이 달라진다. 본 논문에서 제안된 기법은 기존의 FRM과 Dual-Match에 모두 적용될 수 있는 기법이며, 검색 성능을 크게 향상시킬 수 있다.

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 가능한한 적은 수의 인덱스들을 선택하여 구성하는 것이 대단히 중요하다.

본 논문의 주요 공헌은 다음과 같다.

(1) 서브시퀀스 매칭을 수행하는 데에 있어 질의 시퀀스 길이와 윈도우 크기 간의 차이로 인한 성능의 변화를 사전 실험을 통하여 살펴본다.
(2) 윈도우 크기 효과에 의한 성능 저하를 개선하기 위해 인덱스 보간법에 기반한 새로운 검색 기법을 제안한다.
(3) 항목 (1)의 사전 실험 결과에 기반하여 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.
(4) 실제 데이터와 합성 데이터를 이용한 여러 가지 실험을 통하여 제안된 기법으로 인한 성능 개선 효과를 정량적으로 검증한다.

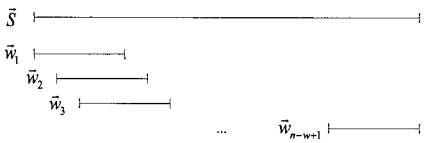
본 논문의 구성은 다음과 같다. 제2장에서는 관련 연구로서 기존의 서브시퀀스 매칭 기법들을 소개하고 장단점을 논의한다. 제3장에서는 윈도우 크기와 질의 시퀀스 길이 간의 차이와 검색 성능 간의 관계를 규명하는 사전 실험 결과에 관하여 논의한다. 제4장에서는 인덱스 보간법에 기반한 새로운 검색 기법을 제안하고, 제안된 검색 기법의 성능을 최적화할 수 있도록 다수의 인덱스들을 구성하는 알고리즘을 제시한다. 제5장에서는 제안된 기법의 우수성을 여러 가지 실험을 통하여 검증한다. 마지막으로, 제6장에서는 본 논문을 요약하고 결론을 내린다.

2. 관련 연구

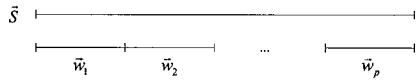
본 장에서는 기존의 서브시퀀스 매칭 기법들에 대하여 간략하게 소개한다. 제2.1절과 제2.2절에서 각각 FRM [8]과 Dual-Match [12]에 대하여 설명한다.

2.1 FRM

FRM [8]은 참고문헌 [1]에서 제시되었던 전체 매칭 기법을 확장하여 제안된 서브시퀀스 매칭 알고리즘이다. FRM은



(a) 슬라이딩 윈도우의 추출.



(b) 디스조인트 윈도우의 추출.

(그림 1) 슬라이딩 윈도우와 디스조인트 윈도우의 추출

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. (그림 1)은 하나의 시퀀스 S로부터 슬라이딩 윈도우와 디스조인트 윈도우를 추출하는 방법을 보인 것이다. 인접한 두 슬라이딩 윈도우들 간의 시작 위치의 오프셋 차는 1이지만, 인접한 디스조인트 윈도우들 간의 시작 위치의 오프셋 차는 w이다.

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[1]. 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 자료구조로서 다차원 인덱스(multidimensional index)를 사용한다. FRM과 Dual-Match에서는 모두 대표적인 다차원 인덱스의 하나인 R*-트리 [3]를 사용한다.

FRM의 서브시퀀스 매칭 단계는 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.

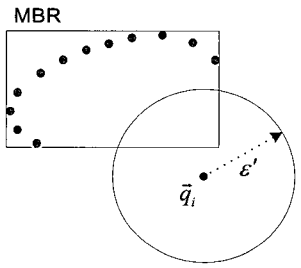
$$\varepsilon' = \frac{\varepsilon}{\sqrt{p}} \left(p = \left\lfloor \frac{Len(Q)}{w} \right\rfloor \right) \qquad (3)$$

1) 일반적으로, 다차원 인덱스를 이용한 탐색 성능은 인덱스의 차원이 증가할수록 기하급수적으로 저하되는데, 이러한 문제를 고차원 문제(high dimensionality problem)라고 한다. DFT 변환의 목적은 고차원 공간 상의 객체를 저차원 공간 상의 객체로 변환함으로써 고차원 문제를 해결하기 위한 것이다. DFT 변환을 이용하여 서브시퀀스 매칭을 수행하여도 착오 기각(false dismissal)이 발생하지 않음을 참고문헌 [8, 12]에서 증명하였다.

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. 참고문헌[8]에서는 최종적으로 반환되어야 할 서브시퀀스들 중 일부를 검색하지 못하는 현상인 착오 기각(false dismissal)이 FRM에서 발생하지 않음을 증명하였다.

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. 실제로 참고문헌[8]에서 모든 데이터 윈도우 점들을 인덱스에 저장한 경우 순차 검색(sequential scan)보다도 성능이 저하된다고 기술하고 있다. 참고문헌 [8]에서는 이러한 문제를 해결하기 위해 윈도우 점들을 개별적으로 인덱스에 저장하지 않고, (그림 2)에서 보는 바와 같이 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 심각한 단점을 갖고 있다[12]. (그림 2)에서 q_i는 질의 시퀀스 Q로부터 추출된 하나의 디스조인트 윈도우이며, ε'은 공식 (3)에서 주어진 값이다. 그림에서 보는 바와 같이 MBR 내의 데이터 윈도우 점들은 실제로 q_i를 중심으로 하는 질의 영역 내에 포함되어 있지 않지만, MBR이 질의 영역과 겹치기 때문에 그 MBR 내의 모든 윈도우 점들과 대응되는 서브시퀀스들이 후보 집합에 포함되게 된다. 이러한 점들은 모두 후보 집합의 크기를 크게 증가시켜 결국 검색 성능을 저하시킨다.



(그림 2) FRM에서의 MBR 구성 및 영역 탐색.

2.2 Dual-Match

Dual-Match는 앞에서 설명한 FRM의 단점을 개선하고자 참고문헌[12]에서 제안한 새로운 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 FLOOR($Len(S)/w$)개이며[2], 길이 $Len(Q)$인 질의 시퀀스 Q로부터 추출되는 윈도우는 모두 ($Len(Q)-w+1$)개이다. 따라서, Dual-Match는 FRM에 대한 이원적(dual) 기법이라 할 수 있다[12].

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, FLOOR((min($Len(Q)$)+1)/2)를 윈도우의 크기 w로 사용한다. 참고문헌[12]에서는 이러한 크기의 윈도우를 사용할 때 Dual-Match가 착오 기각을 발생시키지 않음을 증명하였다. 또한, 다양한 실험을 통하여 Dual-Match의 검색 성능이 FRM과 비교하여 크게 개선됨을 보였다.

3. 연구 동기

이 장에서는 질의 시퀀스 길이와 윈도우 크기 간의 차이가 서브시퀀스 매칭 성능에 크게 영향을 미치는 것을 사전 실험을 통하여 관찰한다. 제3.1절에서는 윈도우 크기 효과에 대하여 설명하고, 제3.2절과 제3.3절에서는 사전 실험을 통해 질의 시퀀스 길이와 윈도우 크기를 변화시키며 서브시퀀스 매칭 실행 시간을 측정한다. 또한, 그 결과 윈도우 크기와 질의 시퀀스 길이 간의 차이와 검색 성능 간의 관계를 규명한다.

3.1 윈도우 크기 효과

FRM 및 Dual-Match에서는 주어진 질의 시퀀스에 대하여 사용되는 윈도우의 크기가 작을수록 착오 채택이 증가하는 경향이 있다. 착오 채택의 증가는 후처리 시에 처리해야 할 대상이 많아짐을 의미하므로 후처리 단계의 수행 시간이 증가하며, 이는 전체 서브시퀀스 매칭 시간 증가의 주요 원인이 된다. 이러한 현상을 윈도우 크기 효과(window size effect)라 한다[12]. 따라서, 가급적 큰 윈도우들을 대상으로 인덱스를 구성하는 것이 서브시퀀스 매칭을 처리하는데 효과적이다.

반면, 인덱스에 저장된 윈도우의 크기가 질의 시퀀스의 길이보다 크다면, 이 인덱스는 해당 질의시퀀스를 위한 서브시퀀스 매칭에서 사용될 수 없다[8, 12]. 이러한 경우에는 순차 검색(sequential search)을 통하여 서브시퀀스 매칭을 수행해야 하므로 처리 시간이 매우 길어진다. 따라서, 서브시퀀스 매칭을 효율적으로 수행하기 위하여 다차원 인덱스에 저장될 윈도우의 크기를 올바르게 선택하는 전략이 필수적이다.

기존의 서브시퀀스 매칭 기법에서 인덱스 구성을 위한 윈도우 크기는 해당 응용에서 사용되는 최소의 질의 시퀀스의 길이에 기반하여 정해지며, FRM에서는 min($Len(Q)$)[8] 그리고 Dual-Match에서는 FLOOR((min($Len(Q)$)+1)/2)[12]가 된다. 그러나, 실제 응용에서 입력되는 질의 시퀀스의 길이는 설정된 윈도우의 크기에 관계없이 매우 다양하므로, 윈도우 크기와 질의 시퀀스 길이의 차가 큰 경우에는 윈도우 크기 효과에 따라 전체 검색 성능이 떨어지는 현상이 발생한다.

본 장에서는 사전 실험을 통하여 윈도우 크기 효과가 서브시퀀스 매칭의 성능에 얼마나 큰 영향을 미치는지에 대하여 고찰한다. 또한, 다음 장에서는 이 사전 실험의 결과 분석을 토대로 윈도우 크기 효과를 고려하여 서브시퀀스 매칭의 성능을 개선시키기 위한 방법을 제시한다.

3.2 사전 실험 설정

사전 실험에서 사용된 데이터는 길이가 1024인 620 개의 한국의 실제 주식 데이터 시퀀스를 사용하였다. 질의 시퀀스는 임의로 선택된 데이터 시퀀스 내의 임의 위치로부터 서브시퀀스를 추출하여 각 요소 값에 적절한 범위 내의 임의의 값을 더하는 방식으로 변형하여 생성하였다. 하드웨어 및 소프트웨어 환경, 데이터 시퀀스의 윈도우 분할 및 저차원 변환, R*-트리의 구성 등에 대해서는 성능 평가 실험을 다루는 제5장에서 구체적으로 설명한다.

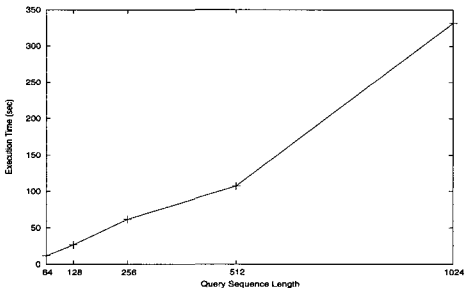
사전 실험은 두 가지로 구분하여 실행하였다. 첫번째 실험은 고정된 크기의 윈도우로 구성된 하나의 인덱스를 이용하여 여러 가지 길이의 질의 시퀀스들에 대한 서브시퀀스 매칭을 수행하여 질의 시퀀스의 길이에 따른 성능 변화를 살펴본다. 첫번째 실험에서 사용된 윈도우 크기는 $w=64$이며, 질의 시퀀스의 길이는 $Len(Q)=64, 128, 256, 512, 1024$이다. 두번째 실험은 첫번째 실험과 반대로 고정된 길이의 질의 시퀀스에 대하여 다양한 크기의 윈도우들로 구성된 각각의 인덱스를 이용하여 서브시퀀스 매칭을 수행함으로써 윈도우의 크기에 따른 검색 성능 변화를 살펴본다. 두번째 실험에서 사용된 질의 시퀀스 길이는 $Len(Q)=1024$이며, 윈도우의 길이는 $w=64, 128, 256, 512, 1024$이다.

성능 지수로는 전체 실행 시간을 사용하였으며, 두 가지 사전 실험 모두에 대하여 허용치 ε은 최종 질의 결과로서 20 개의 서브시퀀스들을 반환하도록 조정하였다.

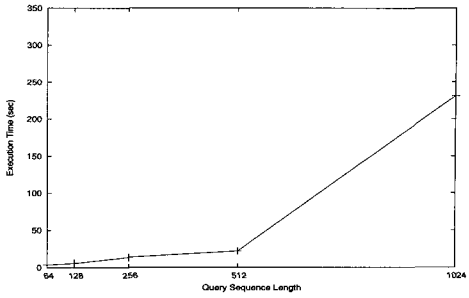
3.3 사전 실험 결과 및 분석

(그림 3) (a)와 (b)는 각각 FRM과 Dual-Match에 대한 첫

2) FLOOR(x)는 x보다 작거나 같은 최대의 정수를 의미한다

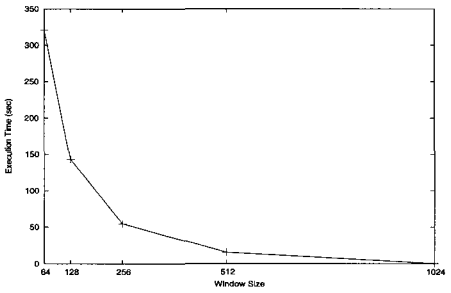


(a) FRM

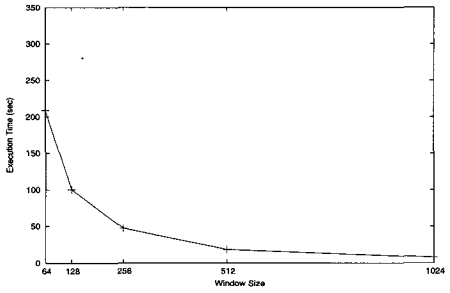


(b) Dual-Match

(그림 3) 질의 시퀀스의 길이에 따른 실행 시간의 변화



(a) FRM



(b) Dual-Match

(그림 4) 윈도우 크기에 따른 실행 시간의 변화

번째 사전 실험 결과를 보인 것이다. 그림 3에서 가로 축은 질의 시퀀스의 길이를 나타내고, 세로 축은 전체 실행 시간을 나타낸다. 그림에 나타난 값들의 단위는 초(second)이다.

첫번째 사전 실험의 결과 FRM과 Dual-Match 모두 질의 시퀀스의 길이가 길어짐에 따라 검색 시간이 점차 증가하였다. 이러한 원인은 윈도우 크기 효과에 따라 질의 시퀀스와 윈도우 크기 간의 차이가 커지면서 인덱스 검색 결과 후보 집합에 포함되는 후보 서브시퀀스의 개수가 증가하기 때문이다.

(그림 4) (a)와 (b)는 각각 FRM과 Dual-Match에 대한 두번째 사전 실험 결과를 보인 것이다. 가로 축은 인덱스를 구성한 윈도우의 크기를 나타내고 세로축은 전체 실행 시간을 나타낸다. 그림에 나타난 값들의 단위는 초이다.

두번째 사전 실험의 결과도 첫번째 실험과 유사하게 얻어졌다. 두번째 사전 실험의 결과 FRM과 Dual-Match 모두 윈도우 크기가 증가함에 따라 검색 시간이 급격하게 감소하였다. 이는 질의 시퀀스와 윈도우 크기 간의 차이가 작아짐에 따라 윈도우 크기 효과에 따라 인덱스 검색 결과 후보 집합에 포함되는 후보 서브시퀀스의 개수가 급격하게 감소하기 때문이다.

두 가지 사전 실험의 결과에 의하여 서브시퀀스 매칭의 성능은 질의 시퀀스의 길이에 대체로 비례하며 윈도우 크기에 반비례한다는 점을 알 수 있다. 이러한 특성에 따라 서브시퀀스 매칭의 실행 비용 T는 아래의 공식 (4)와 같이 표현할 수 있다:

$$T = c\frac{Len(Q)}{w} \quad (c > 0) \tag{4}$$

여기에서 c는 양의 상수(constant)이다.

다음 장에서는 이러한 윈도우 크기 효과를 최소화하여 서브시퀀스 매칭 성능을 크게 향상시킬 수 있는 새로운 기법을 제안한다. 제안된 기법은 앞에서 주어진 공식 (4)를 이용하여 질의 시퀀스들의 길이의 분포를 반영하는 검색 비용 공식을 산출하고, 그 비용 공식에 따라 효율적인 서브시퀀스 매칭을 수행할 수 있도록 최적의 다수의 인덱스를 구성하는 알고리즘을 제시한다.

4. 제안하는 기법

본 장에서는 서브시퀀스 매칭 수행 시에 윈도우 크기 효과에 의한 착오 채택을 줄이기 위하여 인덱스 보간법[10, 11]을 이용하는 새로운 기법을 제안한다. 인덱스 보간법은 서로 다른 윈도우 크기를 이용하는 하나 이상의 인덱스들을 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 가능한한 적은 수의 인덱스들을 선택하여 구성하는 것이 대단히 중요하다. 본 장에서는 물리적 데이터베이스 설계 관점에서 효율적인 서브

시퀀스 매칭을 지원하기 위하여 최적의 윈도우 크기들을 선택하여 인덱스를 구축하는 방안에 대하여 논의한다.

제4.1절에서는 제안하는 기법의 기본 아이디어에 대하여 설명하고, 제안하는 기법의 견고성(robustness)과 효율성에 관하여 논의한다. 제4.2절에서는 서브시퀀스 매칭에 대한 비용 공식과 이 공식을 기반으로 하여 최적의 서브시퀀스 매칭을 지원하기 위한 윈도우 크기들을 선정하는 알고리즘을 제안한다.

4.1 기본 아이디어

기존의 서브시퀀스 매칭 알고리즘에서의 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. 실제 예로, 주식 데이터베이스의 경우 중기적 패턴 분석을 위해서는 20일을 전후하여, 단기적으로는 5일을 전후하여 질의 시퀀스를 생성하고, 장기적인 패턴 분석의 경우 몇 년간의 변화 추이에 대하여 검색을 수행할 수도 있다. 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 보간법 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:

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

$$w_{max} = \max\{w_i \mid w_i \le \lfloor (Len(Q)+1)/2 \rfloor\ (1 \le i \le k)\}$$

$$\text{(Dual-Match)} \qquad (5)$$

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 견고성을 보이기 위한 것이다.

보조 정리 1. 공식 (5)에 의해 선택된 w-인덱스를 이용하여 수행한 서브시퀀스 매칭에서는 착오 기각이 발생하지 않는다.

증명 : 입력으로 주어질 수 있는 모든 질의 시퀀스 Q를 공식 (5)에 의해 같은 w_{max} 값을 갖는 k 개의 그룹 $Q_1, Q_2, \cdots, Q_k$로 분할할 수 있다. 모든 질의 시퀀스는 $Q_1, Q_2, \cdots, Q_k$ 중의 오직 하나의 그룹에 반드시 포함된다. 각 질의 시퀀스 그룹 Q_i에 포함된 모든 질의 시퀀스 Q는 공식 (5)에 의해 $w_{max} \le Len(Q)$ 또는 $w_{max} \le \text{FLOOR}((Len(Q) + 1) / 2)$인 관계가 성립한다.

참고문헌[8, 12]에서는 FRM과 Dual-Match에서 하나의 윈도우 크기 w에 대한 인덱스를 생성하였을 때에 착오 기각이 발생하지 않음을 이미 증명한 바 있다. 여기에서, 두 기법에 있어서 착오 기각을 유발시키지 않기 위한 유일한 조건은 윈도우의 크기 w와 모든 질의 시퀀스의 길이 $Len(Q)$에 대하여 $w \le Len(Q)$ 또는 $w \le \text{FLOOR}((Len(Q)+1)/2)$인 관계가 성립해야 한다는 것이며, 다른 어떠한 조건도 추가로 요구하고 있지 않다.

따라서, 각 질의 시퀀스 그룹 Q_i에 포함된 질의 시퀀스 Q는 공식 (5)에 의해 선택된 w-인덱스를 이용하여 서브시퀀스 매칭을 수행할 때, 착오 기각이 발생하지 않는다. 결론적으로, 모든 입력 가능한 질의 시퀀스에 대하여 공식 (5)에 의해 선택된 w-인덱스를 이용하여 서브시퀀스 매칭을 수행할 때, 착오 기각이 전혀 발생하지 않는다. □

위의 공식 (5)에 의해 선택된 윈도우 크기 w_{max}에 대한 w-인덱스를 이용하는 것이 다른 윈도우 크기 w' ($\neq w_{max}$)에 대한 w-인덱스를 이용하는 것보다 대체로 더 나은 성능을 얻을 수 있다. 만약 w'이 w_{max}보다 작다면 $w' < w_{max} \le Len(Q)$ (FRM의 경우) 또는 $w' < w_{max} \le \text{FLOOR}((Len(Q) + 1) / 2)$ (Dual-Match의 경우)인 관계가 성립한다. 윈도우 크기 효과에 의하여 w' 크기의 윈도우로 구축한 인덱스를 이용한 검색 성능은 w_{max} 크기의 윈도우로 구축한 인덱스를 이용한 검색 성능보다 떨어지는 경향이 있다.

반면에, 만약 w'이 w_{max}보다 크다면 $w' > Len(Q)$ (FRM의 경우) 또는 $w' > \text{FLOOR}((Len(Q)+1)/2)$ (Dual-Match의 경우)인 질의 시퀀스 Q가 반드시 존재한다. FRM과 Dual-Match 모두 질의 시퀀스 Q에 대하여 w' 크기의 윈도우로 구축된 인덱스를 이용하여 검색을 수행할 수 없으며, 반드시 순차 검색을 수행하여야 한다. 순차 검색의 성능은 인덱스를 이용한 검색에 비하여 성능이 크게 떨어진다.

4.2 다중 인덱스의 구성 방안

앞 절에서는 이미 생성된 다수의 인덱스 중에서 주어진 질의 시퀀스에 가장 적합한 인덱스를 선택하여 검색을 수행하는 방법에 대하여 설명하였다. 이 절에서는 응용에서 사용될 전체 질의 시퀀스들을 대상으로 서브시퀀스 매칭의 성능을 최대화하기 위하여 w-인덱스들을 구축하는 방법에 대하여 설명한다.

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 $w_1, w_2, \cdots, w_k$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 논문에서는 제5장에서 실험을 통하여 이를 규명한다.

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 $Q_1, Q_2, \cdots, Q_g$ ($Len(Q_1) < Len(Q_2) < \cdots < Len(Q_g)$)로 분할한다. 각 그룹에 대하여 공식 (5)에 의하여 선택된 윈도우의 크기를 $w_{max}(Q_1), w_{max}(Q_2), \cdots, w_{max}(Q_g)$ ($w_{max}(Q_1) \le w_{max}(Q_2) \le \cdots \le w_{max}(Q_g)$)라 한다. 여기에서, 인덱스의 개수 k는 질의 시퀀스 그룹의 개수 g보다 작거나 같다고 가정한다.

이러한 설정 하에 제3장에서 주어진 공식 (4)를 확장하여 인덱스 보간법에 기반하고 전체 질의 시퀀스들을 대상으로 서브시퀀스 매칭을 수행하기 위한 전체 검색 비용 T를 구하면 아래의 공식 (6)과 같다[3]:

$$T = \sum_{1 \le i \le g} \left(\frac{Len(Q_i)}{w_{\max}(Q_i)} \right) \cdot F_i \qquad (6)$$

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

3) 공식 (4)에서와 달리 공식 (6)에서 양의 상수를 포함하지 않은 이유는 공식 (6)에 의하여 구해진 비용 값이 (그림 5)의 윈도우 크기 선택 알고리즘에서 상대적으로 비교되기만 하기 때문이다.

```
Procedure GetWindowSizes
(1)   Compute w_max(Q_1);
(2)   for i = 2 .. k do
(3)       for each possible w other than w_max(Q_j)
          (1≤j < i) do
(4)           Compute T;        // using Eq. (6)
(5)           if T is minimum then
(6)               w_max(Q_i) = w;
(7)           end if
(8)       end for
(9)   end for
end.
```

(그림 5) 윈도우 크기 선정 알고리즘.

공식 (6)의 비용 T를 최소로 하는 윈도우 크기 $w_{max}(Q_i)$ ($1 \le i \le k$)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 5)에 보인 알고리즘은 주어진 질의 시퀀스들의 길이 분포와 w-인덱스의 개수 k를 입력으로 받아, 공식 (6)에 보인 비용 T 값을 최소화함으로써 전체 서브시퀀스 매칭의 성능을 최대화하기 위한 윈도우 크기 $w_{max}(Q_i)$ ($1 \le i \le k$)를 반환한다. 이 알고리즘은 두 개의 중첩된 for 루프만을 포함하고 있으므로, 실행 시간 복잡도는 $O(M \cdot k)$이다. 이 알고리즘에서 공식 (6)을 계산하기 위한 $Len(Q_i)$, F_i, g 값은 전역으로 액세스 가능하다고 가정한다.

(그림 5)의 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 $w_{max}(Q_i)$ ($2 \le i \le k$)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 $w_{max}(Q_j)$ ($1 \le j < i$)를 제외한 나머지 윈도우 크기 w를 검토 대상으로 한다. 라인 (4)에서는 이전에 정해진 $w_{max}(Q_j)$ ($1 \le j < i$)와 윈도우 크기 w를 공식 (6)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.

5. 성능 평가

이 장에서는 실험에 의한 성능 분석을 통하여 제안된 기

법의 우수성을 보인다. 먼저, 제5.1절에서는 실험 환경을 설명하고, 제5.2절에서는 기존 기법과의 비교 실험을 통해 제안된 기법의 성능 개선 효과를 정량적으로 보인다.

5.1 실험 환경

본 실험에서는 성능 평가를 위하여 실제 데이터와 합성 데이터를 이용하였다. 실제 데이터는 사전 실험에서 사용하였던 한국 주식 데이터로, 길이가 1024인 620개의 데이터 시퀀스들로 구성된다. 합성 데이터는 길이 1024인 1000개, 3000개, 5000개의 데이터 시퀀스로 구성되며 다음과 같은 랜덤 워크(random walk) 공식에 의하여 생성되었다:

$$s_{i+1} = s_i + z_i \quad (i \geq 1) \tag{7}$$

여기에서, s_1은 영역[1, 10] 내의 임의의 값이며, z_i는 [-0.1, 0.1] 내에서 임의로 선택한 값이다.

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 아래의 <표 1>과 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 20개의 서브시퀀스를 반환하도록 조정하였다.

〈표 1〉 그룹 내에 포함된 질의 시퀀스의 개수.

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

실험을 위한 환경으로는 1.8 GHz Pentium 4 프로세서와 512 MB의 메모리를 장착한 PC와 MS Windows 2000 운영체제를 사용하였다. 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.

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

5.2 실험 결과

본 논문에서는 세 가지 실험을 수행한다. 첫째, 실제 데이터를 이용하여 w-인덱스의 개수를 변경하며 방법 (A), (B), (C) 간의 성능을 비교한다. 둘째, 합성 데이터를 이용하여 데이터 시퀀스의 개수를 변경하면서 성능을 비교한다. 셋째, 합성 데이터를 이용하여 w-인덱스의 개수를 변경하며 성능을 비교한다.

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

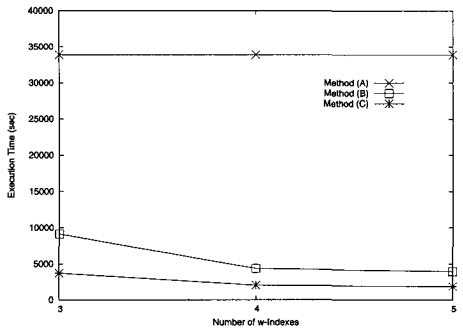
(그림 6)은 첫번째 실험의 결과를 보인 것이다. 그림 6(a)는 FRM에 대하여, 그림 6(b)는 Dual-Match에 대하여 실험한 결과이다. (그림 6)에서 가로 축은 w-인덱스의 개수, 세로 축은 실행 시간을 초 단위로 나타낸 것이다.

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 6) (a)에서 5 개의 w-인덱스를 이용한 경우 방법 (C)는 방법 (A)에 비하여 약 18.4배, 방법 (B)에 비하여 약 2.1배 성능이 향상되었다. (그림 6) (b)에서 5개의 w-인덱스를 이용한 경우 방법 (C)는 방법 (A)에 비하여 약 13.3배, 방법 (B)에 비하여 약 1.8배 성능이 향상되었다.

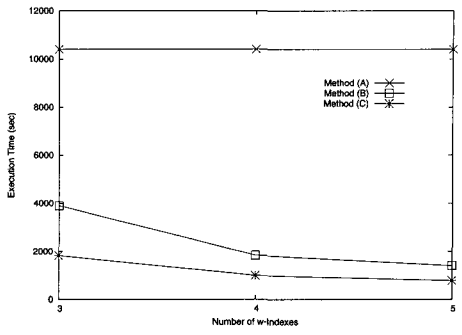
두번째 실험은 확장성(scalability)에 대한 실험으로, 첫번째 실험과 마찬가지로 FRM과 Dual-Match에 대하여 개별적으로 수행하였으며, 합성된 데이터 시퀀스의 개수를 1000, 3000, 5000 개로 증가하면서 성능 변화를 관찰하였다. 두 기법 모두 방법 (B)와 방법 (C)에서 5개의 w-인덱스들을 사용하였고, 첫번째 실험에서 인덱스의 개수가 5인 경우와 같은 w-인덱스들을 사용하였다.

(그림 7)은 두번째 실험의 결과를 보인 것이다. (그림 7) (a)는 FRM에 대하여, (그림 7) (b)는 Dual-Match에 대하여 실험한 결과이다. (그림 7)에서 가로 축은 데이터 시퀀스의 개수, 세로 축은 실행 시간을 초 단위로 나타낸 것이다.

두 가지 알고리즘 모두에 대하여 데이터 시퀀스의 개수가 증가하더라도 첫번째 실험 결과와 유사하게 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 나은 성능을 보였다. (그림 7) (a)에서 5000 개의 데이터 시퀀스에 대하여 실험한 경우 방법 (C)는 방법 (A)에 비하여 약 35.5배, 방법 (B)에 비하여 약 3.5배 성능이 향상되었다. (그림 7) (b)에서 5000 개의 데이터 시퀀스의 경우 방법 (C)는 방법 (A)에 비하여 약 14.5배, 방법 (B)에 비하여 약 2.1배 성능이 향상되었다.

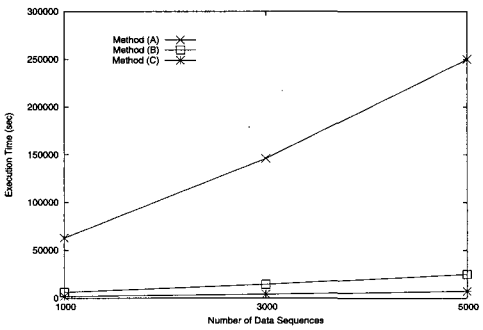


(a) FRM

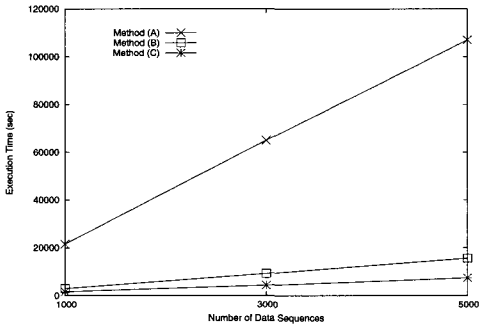


(b) Dual-Match

(그림 6) w-인덱스의 개수에 의한 성능 비교



(a) FRM



(b) Dual-Match.

(그림 7) 데이터 시퀀스 개수에 의한 성능 비교.

이러한 성능 향상 비율은 데이터 시퀀스의 개수에 관계 없이 대체로 비슷하며, 첫번째 실험에서의 결과와도 유사하다.

세번째 실험은 3000개의 합성 데이터 시퀀스에 대하여 w-인덱스의 개수를 3, 4, 5개로 변경하면서 세 가지 방법 (A), (B), (C)의 성능의 변화를 살펴보았다. 앞의 두 실험에서와 마찬가지로 FRM과 Dual-Match에 대하여 개별적으로 수행하였다. 이 실험을 통하여 첫번째 실험과 매우 유사한 결과를 얻었으며, 결과 그래프도 거의 동일한 모양으로 형성되었다. 즉, FRM과 Dual-Match 모두에 대하여 w-인덱스의 개수가 증가함에 따라 성능이 향상되었으며, 기존의 방식을 그대로 적용한 방법 (A)와 일정 간격의 w-인덱스를 이용한 방법 (B)에 비하여 제안된 기법에 의하여 정해진 w- 인덱스를 이용한 방법 (C)가 큰 성능 개선 효과를 가지는 것으로 나타났다.

두번째와 세번째 실험에서 살펴본 바와 같이 합성 데이터를 이용한 실험에서도 본 논문에서 제안된 기법이 기존의 기법들에 비하여 더욱 좋은 성능을 보였으며, 제안된 기법의 뛰어난 성능 개선 효과를 규명할 수 있었다.

5. 결 론

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. 먼저, 사전 실험을 통하여 서브시퀀스 매칭을 수행하는 데에 있어 질의 시퀀스 길이와 윈도우 크기 간의 차이로 인한 성능의 변화를 살펴보았고, 사전 실험의 결과에 기반하여 물리적 데이터베이스 설계 관점에서 서브시퀀스 매칭 처리를 위한 검색 비용 공식을 산출하였다. 다음에, 윈도우 크기 효과에 의한 성능 저하를 개선하기 위해 인덱스 보간법에 기반한 새로운 검색 기법을 제안하였고, 검색 비용 공식에 기반하여 제안된 검색 기법의 성능을 최적화할 수 있도록 인덱스를 구성하는 알고리즘을 제시하였다. 마지막으로, 실제 데이터와 합성 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 18.4~35.5배의 성능 향상을 보였으며, 일정 간격의 인덱스를 사용한 경우와 비교하여 약 2.1~3.5배의 성능 향상을 보였다. Dual-Match의 경우, 단일 인덱스를 사용한 경우와 비교하여 약 1.33~14.5배, 일정 간격의 인덱스를 사용한 경우와 비교하여 약 1.8~2.1배의 성능 향상을 보였다.

인덱스 보간법에 기반한 검색 기법의 성능을 최적화하도록 w-인덱스의 개수 k를 설정하는 것이 매우 중요하다. 이 값은 질의 처리 성능뿐만 아니라, w-인덱스에 데이터를 삽

입, 삭제, 갱신하기 위한 관리 비용을 함께 고려하여야 한다. 하지만, w-인덱스의 관리 비용은 전체 데이터베이스의 크기, 데이터의 갱신 주기, 동시 액세스 트랜잭션의 개수, 물리적 공간의 크기 등의 여러가지 인자들을 복합적으로 고려하여야 한다. 이러한 w-인덱스의 관리 비용에 대한 연구는 추후 별도의 연구 과제로 수행할 것이다.

참 고 문 헌

[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.

노 웅 기

e-mail : woong@mozart.kaist.ac.kr

1991년 한국과학기술원 전산학과 학부과정

1993년 한국과학기술원 전산학과(석사, 멀티미디어 전공)

2001년 한국과학기술원 전산학과(박사, 데이터 마이닝 전공)

1995년 일본 동경 NHK 방송기술연구소 방문(일한재단 초청)

1997년 미국 캘리포니아 Stanford 대학교 전산학과 방문 (Prof. Gio Wiederhold 초청)

2001년~2003년 ㈜티맥스소프트 책임연구원 (IBM Mainframe 마이그레이션 솔루션 개발)

2003년~2005년 ㈜티맥스데이타 수석연구원 (DBMS 개발)

2005년 현재 한국과학기술원 전산학과 초빙교수

관심분야 : 데이터 마이닝, 정보 검색, 멀티미디어, 메인 메모리 DB, 임베디드 DB, Bioinformatics

김 상 욱

email : 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

1999년~2000년 미국 IBM T.J. Watson Research Center Post-Doc.

1995년~2000년 강원대학교 컴퓨터정보통신공학부 부교수

2003년~현재 한양대학교 정보통신대학 정보통신학부 부교수

관심분야 : 데이타베이스 시스템, 저장 시스템, 데이타 마이닝, 멀티미디어 정보 검색, 공간 데이터베이스/GIS, 주기억장치 데이타베이스, 트랜잭션 관리