시계열 서브시퀀스 매칭을 위한 최적의 다중 인덱스 구성 방안

(Optimal Construction of Multiple Indexes for Time-Series Subsequence Matching)

임 승 환 † 김 상 욱 †† 박 희 진 ††
(Seung-Hwan Lim) (Sang-Wook Kim) (Heejin Park)

요 약 일정 기간 동안 객체의 변화한 값들을 기록한 것을 그 객체에 대한 시계열 데이타 시퀀스라고 부르며, 이들의 집합을 시계열 데이타베이스라고 한다. 서브시퀀스 매칭은 주어진 질의 시퀀스와 변화의 추세가 유사한 서브시퀀스들을 시계열 데이타베이스로부터 검색하는 연산이다. 본 논문에서는 서브시퀀스 매칭의 성능을 극대화하기 위한 방안을 제시한다. 먼저, 윈도우 크기 효과로 인한 서브시퀀스 매칭의 심각한 성능 저하 현상을 정량적으로 관찰하여, 하나의 윈도우 크기를 대상으로 만든 단 하나의 인덱스만을 이용하는 것은 실제 응용에서 만족할만한 성능을 제공할 수 없다는 것을 규명하였다. 또한, 이러한 문제로 인해 다양한 윈도우 크기들을 기반으로 다수의 인덱스들을 구성하여 서브시퀀스 매칭을 수행하는 인덱스 보간법의 응용이 필요함을 보였다. 인덱스 보간법을 응용하여 서브시퀀스 매칭을 수행하기 위해서는 먼저 다수의 인덱스들을 위한 윈도우 크기들을 결정해야 한다. 본 연구에서는 물리적 데이타베이스 설계 방식을 이용하여 이러한 최적의 다수의 윈도우 크기들을 선정하는 문제를 해결하였다. 이를 위하여 시계열 데이타베이스에서 수행될 예정인 질의 시퀀스들의 집합과 인덱스 구성의 기반이 되는 윈도우들의 크기의 집합이 주어질 때, 전체 서브시퀀스 매칭들을 수행하는 데에 소요되는 비용을 예측할 수 있는 공식을 산출하였다. 또한, 이 비용 공식을 이용하여 전체 서브시퀀스 매칭들의 성능을 극대화 할 수 있는 최적의 윈도우 크기들을 결정하는 알고리즘을 제안하였으며, 이 알고리즘의 최적성과 효율성을 이론적으로 규명하였다. 끝으로, 실제 주식 데이타와 대량의 합성 데이타를 이용한 실험 결과, 제안된 기법은 기존의 단순한 기법과 비교하여 1.5배에서 7.8배 성능이 향상됨을 보였다.

키워드 : 시계열 데이타베이스, 서브시퀀스 매칭, 인덱스 보간법, 물리적 데이타베이스 설계

Abstract A time-series database is a set of time-series data sequences, each of which is a list of changing values of the object in a given period of time. Subsequence matching is an operation that searches for such data subsequences whose changing patterns are similar to a query sequence from a time-series database. This paper addresses a performance issue of time-series subsequence matching. First, we quantitatively examine the performance degradation caused by the *window size effect*, and then show that the performance of subsequence matching with a single index is not satisfactory in real applications. We argue that *index interpolation* is fairly useful to resolve this problem. The index interpolation performs subsequence matching by selecting the most appropriate one from multiple indexes built on windows of their inherent sizes. For index interpolation, we first decide the sizes of windows for multiple indexes to be built. In this paper, we solve the problem of selecting optimal window sizes in the perspective of *physical database design*. For this, given a set of query sequences to be performed in a target time-series database and a set of window sizes for building multiple indexes, we devise a formula that estimates the cost of all the subsequence matchings. Based on this formula, we propose an algorithm that determines the optimal window sizes for maximizing

· 이 논문은 정보통신부 IT 연구센터(텔리매틱스 요소 기술 연구)의 연구비 지원을 받았습니다.
† 학생회원 : 한양대학교 정보통신공학과
firemoon@ihanyang.ac.kr
†† 종신회원 : 한양대학교 정보통신대학 컴퓨터전공 교수
wook@hanyang.ac.kr
hjpark@hanyang.ac.kr
논문접수 : 2005년 4월 13일
심사완료 : 2005년 9월 23일

the performance of entire subsequence matchings. We formally prove the optimality as well as the effectiveness of the algorithm. Finally, we perform a series of extensive experiments with a real-life stock data set and a large volume of a synthetic data set. The results reveal that the proposed approach improves the previous one by 1.5 to 7.8 times.

Key words : time-series databases, subsequence matching, index interpolation, physical database design

1. 서 론

우리 주위에는 주가지수, 환율, 기온과 같이 시간에 따라 값이 변하는 객체들이 존재한다. 이러한 각 객체의 일정 기간 동안 변화한 값들을 기록한 것을 그 객체에 대한 시계열 데이타 시퀀스(data sequence)라고 부른다[1-4]. 예를 들어, 최근 1년 동안 매일 12시에 측정한 서울의 기온 값들의 리스트는 하나의 시계열 데이타 시퀀스에 해당된다. 시계열 데이타 시퀀스들이 저장되어 있는 데이타베이스를 시계열 데이타베이스(time-series database)라 한다[1,2,4,5].

시계열 데이타베이스에서 한 객체에 대해서 과거의 값 변화를 참조하여 미래의 값을 예측하는 것이 가능하다. 예를 들어, 여러 종목들의 10년간의 주가 변화를 저장하고 있는 시계열 데이타베이스가 있다고 가정하자. 우리는 관심 종목과 최근 1주일간의 주가의 변화가 유사한 시퀀스들을 시계열 데이타베이스에서 찾아냄으로써 앞으로 그 종목의 주가가 어떻게 변할지를 예측할 수 있다. 이 과정에서 주어진 시퀀스를 가지고 데이타베이스에서 그와 유사한 시퀀스들을 찾아내는 일을 유사 시퀀스 매칭(similar sequence matching)이라 한다[1,2,4,5]. 유사 시퀀스 매칭은 데이타 마이닝(data mining) 및 데이타 웨어하우징(data warehousing) 분야에서 중요한 연산으로 사용된다[6,7].

유사 시퀀스 매칭은 다음과 같이 전체 매칭(whole matching)과 서브시퀀스 매칭(subsequence matching)으로 구분된다[4].

- 전체 매칭: 데이타베이스 D 내에 존재하는 시퀀스 S_1, S_2, ..., S_N에 대하여 질의 시퀀스 Q와 유사한 시퀀스 S_i를 검색한다. 이때, S_1, S_2, ..., S_N 및 Q의 길이는 동일해야 한다.
- 서브시퀀스 매칭: 데이타베이스 D 내에 존재하는 시퀀스 S_1, S_2, ..., S_N에 대하여 질의 시퀀스 Q와 유사한 서브시퀀스 X를 포함하는 시퀀스 S_i와 이 서브시퀀스 X가 S_i내에서 시작하는 위치를 검색한다. 이때, S_1, S_2, ..., S_N 및 Q는 서로 다른 길이를 갖는 것이 허용된다.

전체 매칭과는 달리, 서브시퀀스 매칭은 사용되는 데이타 및 질의 시퀀스의 길이에 대한 제약이 없으므로 실제 응용 분야에서 널리 사용된다. 따라서 본 논문은 이러한 서브시퀀스 매칭을 연구의 대상으로 한다.

두 시퀀스 간에 유사성을 측정하는 척도로는 아래와 같이 정의되는 유클리드 거리(Euclidean distance)가 널리 사용된다[1,4,7-11].[1] 즉 길이가 n인 서로 다른 두 시퀀스 $X(=\langle x_0, x_1, ..., x_{n-1}\rangle)$와 $Y(=\langle y_0, y_1, ..., y_{n-1}\rangle)$를 각각 n차원 공간상의 한 점으로 매핑하고, 두 점 사이의 유클리드 거리 D(X, Y)가 ε이하이면, 두 시퀀스 X, Y는 ε-매치(ε-match)한다고 한다[5].

$$D(X,Y)=\sqrt{\sum_{i=0}^{n-1}(x_i-y_i)^2}$$

서브시퀀스 매칭을 위한 처리 기법으로 대표적인 두 가지 방법이 있다. 이들은 각각 참고문헌 [4]와 [5]에서 소개된 방법이다. 참고문헌 [5]의 명칭을 따라 본 논문에서는 [4]의 기법을 FRM, [5]의 기법을 Dual-Match라 부른다. FRM과 Dual-Match는 효과적인 서브시퀀스 매칭을 위하여 인덱스를 사용한다. 또한, 인덱싱의 단위로서 윈도우(window)라는 개념을 이용한다. 윈도우란 시퀀스로부터 추출되는 서브시퀀스로서, 사전에 결정된 고정된 길이 w의 크기를 갖는다. 서브시퀀스 매칭을 위하여 두 기법이 취하는 공통적인 아이디어는 다음과 같다.

먼저, 인덱스 구성을 위하여 각 시퀀스로부터 길이 w의 윈도우들을 추출하고, 각 윈도우를 이산 퓨리에 변환(discrete Fourier transform: DFT) 혹은 웨이블릿 변환(wavelet transform)을 이용하여 저차원 f-공간(f << w) 상의 윈도우 점(window point)으로 변환한다. 효과적인 서브시퀀스 매칭을 위하여 이러한 윈도우 점들을 다차원 인덱스(multidimensional index)의 하나인 R^*-트리[17]에 저장한다.

허용치가 ε인 서브시퀀스 매칭의 처리를 위하여 길이 l인 질의 시퀀스로부터 길이 w의 윈도우들을 추출하고, 각 윈도우를 DFT 혹은 웨이블릿 변환(wavelet transform)을 이용하여 저차원 f-공간(f << w) 상의 윈도우 점으로 변환한다. 각 윈도우 점에 대하여 $\varepsilon/\sqrt{p}$ (p =

1) 유사성을 측정하기 위한 척도로서 유클리드 거리 이외에도 응용에 따라 맨하탄 거리(Manhattan distance)[12], 대응되는 요소 쌍의 최대 거리(maximum distance in any pair of elements)[2], 타임 워핑 거리(time warping distance)[13-16] 등이 사용될 수 있다.

⌊l/w⌋)를 허용치로 갖는 범위 질의를 R*-트리 상에서 수행한다. 본 논문에서는 이 과정을 인덱스 검색 단계(index search step)라 부른다. 이러한 인덱스 검색 단계의 결과, 질의 시퀀스와 ε-매치 할 가능성이 높은 많은 후보 서브시퀀스(candidate subsequence)들이 반환된다. 그 다음, 착오 채택(false alarm)[1,4]을 해결하기 위하여 이 후보 서브시퀀스들을 포함하는 각 시퀀스를 디스크로부터 액세스하여 후보 서브시퀀스가 질의 시퀀스와 실제로 ε-매치 하는가의 여부를 판단한다. 본 논문에서는 이 과정을 후처리 단계(post-processing step)라 부른다.

FRM과 Dual-Match에서 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[5], 제3장에서 좀더 상세히 설명한다. FRM과 Dual-Match에서는 최소 질의 시퀀스의 길이를 기준으로 윈도우 크기를 설정하여 단 하나의 인덱스를 구성하고, 이를 이용하여 서브시퀀스 매칭을 수행한다. 그러나 이러한 방법은 질의 시퀀스의 길이가 증가할수록 서브시퀀스 매칭의 성능이 저하되는 문제점을 안고 있다. 이러한 문제를 해결하기 위하여 입력 가능한 모든 질의 시퀀스의 길이에 대하여 각각 인덱스를 생성하는 방법을 생각해 볼 수 있으나, 인덱스 관리 비용이 지나치게 증가하므로 실제 응용에 적용하는 데에는 현실성이 떨어지게 된다.

본 논문에서는 이러한 문제점을 해결하고자 인덱스 보간법(index interpolation)[18]에 기반한 새로운 서브시퀀스 매칭 기법을 제안한다. 인덱스 보간법이란 둘 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, 데이타가 삽입, 삭제, 갱신될 때마다 해당 인덱스를 그에 따라 모두 변경하는 비용을 포함한다. 본 논문에서는 인덱스 보간법에서 최적의 검색 성능을 지원할 수 있는 가능한한 적은 수의 인덱스들을 선택하여 구성하는 방안에 대하여 논의한다.

본 논문의 주요 공헌은 다음과 같다. 먼저, 윈도우 크기 효과로 인한 서브시퀀스 매칭의 심각한 성능 저하 현상을 정량적으로 관찰하여, 하나의 윈도우 크기를 대상으로 만든 단 하나의 인덱스만을 이용하는 것은 실제 응용에서 만족할만한 성능을 제공할 수 없다는 것을 규명하였다. 또한, 이러한 문제로 인해 다양한 윈도우 크기들을 기반으로 다수의 인덱스들을 구성하여 서브시퀀스 매칭을 수행하는 인덱스 보간법의 응용이 필요함을 보였다. 인덱스 보간법을 응용하여 서브시퀀스 매칭을 수행하기 위해서는 먼저 다수의 인덱스들을 위한 윈도우 크기들을 결정해야 한다. 본 연구에서는 물리적 데이타베이스 설계 방식을 이용하여 이러한 최적의 다수의 윈도우 크기들을 선정하는 문제를 해결하였다. 이를 위하여 시계열 데이타베이스에서 수행될 예정인 질의 시퀀스들의 집합과 인덱스 구성의 기반이 되는 윈도우들의 크기의 집합이 주어질 때, 전체 서브시퀀스 매칭들을 수행하는 데에 소요되는 비용을 예측할 수 있는 공식을 산출하였다. 또한, 이 비용 공식을 이용하여 전체 서브시퀀스 매칭들의 성능을 극대화 할 수 있는 최적의 윈도우 크기들을 결정하는 알고리즘을 제안하였으며, 이 알고리즘의 최적성과 효율성을 이론적으로 규명하였다. 끝으로, 제안된 기법이 기존의 단순한 기법과 비교하여 상당한 성능 개선 효과가 있음을 다양한 실험을 통해서 검증하였다.

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 관련 연구로서 유클리드 거리 기반의 유사 모델을 사용하는 기존의 유사 시퀀스 매칭 기법들을 소개하고, 그 장단점에 관하여 논의한다.[2)]

2) 응용에 따라 유클리드 거리만을 이용한 시퀀스 매칭을 통해서는 사용자가 원하는 시퀀스들을 검색하지 못하는 경우가 발생할 수 있다. 따라서 응용 분야의 특성에 따라 유사한 정도를 유연하게 정의할 수 있도록 변환(transformation)이 사용된다. 이러한 변환을 함께 지원하는 유사 시퀀스 매칭 기법들이 다음의 참고 문헌들에 제안되어 있다. 정규화(normalization) 지원 기법: [2,7,9,10,21,22], 이동 평균(moving average) 지원 기법: [7,11,18], 타임 워핑(time warping) 지원 기법: [7,13-16]

2.1 전체 매칭

참고문헌 [1]에서는 모든 데이타 시퀀스들과 질의 시퀀스의 길이가 동일하다는 전제 하에 전체 매칭 기법을 제안하였다. 먼저, 길이 l인 각 데이타 시퀀스를 DFT를 이용하여 저차원 f-공간(f << l) 상의 점으로 변환하고, 이들을 R*-트리[17]에 저장함으로써 인덱싱을 수행한다. 여기서, 저차원 변환은 R*-트리 등의 다차원 인덱스(multidimensional index)가 가지는 고차원 문제(dimensionality curse)[1,19,20]를 해결하기 위하여 사용된다.

전체 매칭을 위하여 길이가 l인 질의 시퀀스를 위와 동일한 방법으로 저차원 f-공간상의 한 점으로 변환하고, 변환한 질의 점을 중심점, 허용치 ε을 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, 질의 점과 유클리드 거리가 ε 이하인 모든 데이타 점들과 대응되는 데이타 시퀀스들을 반환하는데, 이들을 후보 집합(candidate set)이라 한다.

질의 시퀀스와 ε-매치 하는 데이타 시퀀스들을 후보 집합에 포함시키지 못하는 현상을 착오 기각(false dismissal)이라 한다. 반면, 질의 시퀀스와 ε-매치 하지 않는 일부의 데이타 시퀀스들을 후보 집합에 포함시키는 현상을 착오 채택(false alarm)이라 한다. 참고 문헌 [1]에서는 Parseval 정리를 이용하여 이 기법이 착오 기각을 유발하지 않음을 증명하였다. 그러나 저차원 변환 시의 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 데이타 시퀀스를 액세스하고, 이 시퀀스가 질의 시퀀스와 실제로 ε-매치 하는가의 여부를 조사함으로써 착오 채택을 제거한 최종 질의 결과를 생성한다.

2.2 FRM

참고문헌 [4]에서는 임의의 길이를 갖는 질의 시퀀스와 데이타 시퀀스들을 대상으로 하는 서브시퀀스 매칭 방법 FRM을 제안하였다. FRM은 전체 매칭 기법의 기본 아이디어를 확장한 것으로서, R*-트리 인덱싱을 위하여 고정된 길이 w를 갖는 윈도우(window) 개념을 이용한다.

먼저, 길이가 Len(S)인 각 데이타 시퀀스 S로부터 모든 가능한 위치에서 시작되는 길이 w의 슬라이딩 윈도우(sliding window)들을 추출하고, DFT를 이용하여 각 윈도우를 저차원 f-공간상(f << w)의 한 점으로 변환한다. 본 논문에서는 이 점을 데이타 윈도우 점(data window point)이라 정의한다. 시퀀스 S로부터 슬라이딩 윈도우들을 추출하므로 각 시퀀스 당 생성되는 데이타 윈도우 점들의 수는 (Len(S)-w+1)이 된다. 이 결과, 인덱싱의 대상이 되는 전체 데이타 윈도우 점들의 수가 매우 많아지므로, R*-트리를 위한 저장 공간이 커지며, 이 결과 인덱스 검색 단계의 성능이 크게 저하된다[4]. FRM에서는 이러한 문제를 해결하기 위하여 다수의 데이타 윈도우 점들을 포함하는 최소 포함 사각형(minimum bounding rectangle: MBR)들을 구성한 후, 이 MBR들을 R*-트리[17]에 저장하는 방법을 사용하였다.

서브시퀀스 매칭을 위해서는 길이 Len(Q)인 질의 시퀀스 Q로부터 길이 w인 $\lfloor Len(Q)/w \rfloor$ 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 질의 윈도우 점을 중심점, 허용치 $\varepsilon/\sqrt{p}$ ($p = \lfloor Len(Q)/w \rfloor$)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 $\varepsilon/\sqrt{p}$ 이하인 데이타 윈도우 점들을 포함한다. 이러한 점들과 대응되는 데이타 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 ε-매치 하는가를 판단하는 후처리 단계를 수행한다. 참고문헌 [4]에서는 이러한 서브시퀀스 매칭 기법에서 착오 기각이 발생하지 않음을 증명하였다.

2.3 Dual-Match

FRM에서는 인덱싱을 위한 저장 공간의 오버헤드를 줄이기 위하여 개별적인 데이타 윈도우 점들 대신 다수의 윈도우 점들을 포함하는 MBR들을 R*-트리 내에 저장한다. 그러나 이러한 MBR 내부에는 죽은 공간(dead space)[17]이 존재하게 되므로, 이로 인하여 후보 서브시퀀스의 착오 채택이 발생되며, 이것은 처리 성능의 저하로 직결된다[5]. 참고문헌 [5]에서는 이러한 문제점을 해결하기 위한 방법으로서 이원성 기반 서브시퀀스 매칭(duality-based subsequence matching: Dual-Match)을 제안하였다.

Dual-Match에서는 길이가 Len(S)인 데이타 시퀀스 S로부터 슬라이딩 윈도우를 추출하고 길이 Len(Q)인 질의 시퀀스 Q로부터 디스조인트 윈도우를 추출하는 FRM과는 반대로 데이타 시퀀스로부터는 디스조인트 윈도우를 추출하고 질의 시퀀스로부터는 슬라이딩 윈도우를 추출하는 방식을 사용한다. 이와 같은 역할 교환을

통하여 Dual-Match는 각 시퀀스로부터 ⌊Len(S)/w⌋ 개(여기서 w는 윈도우의 길이)의 데이타 윈도우 점들만을 추출하게 되므로 R*-트리에 저장할 데이타 윈도우 점들의 수를 FRM의 약 1/w로 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. 참고문헌 [5]에서는 다양한 실험을 통하여 Dual-Match의 검색 성능이 FRM과 비교하여 크게 개선됨을 보였으며, Dual-Match가 착오 기각 없이 서브시퀀스 매칭을 올바르게 수행함을 증명하였다.

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-Match를 이용한 서브시퀀스 매칭에서는 R*-트리에 저장된 윈도우의 크기가 작을수록 착오 채택의 수가 증가하는 경향이 있다. 예를 들어, 두 R*-트리 R1과 R2에 대하여 R1이 R2보다 큰 윈도우를 이용하여 구성되었다고 가정하자. 이때, R2를 이용한 인덱스 검색 단계를 통하여 반환된 후보 집합 S2 내에는 R1을 이용한 인덱스 검색 단계를 통하여 반환된 후보 집합 S1에는 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[5]. 따라서 큰 윈도우를 사용하여 R*-트리를 구성하는 것은 전체 수행 시간을 개선시키는데 좋은 효과가 있다.

반면, 저장된 윈도우의 크기가 질의 시퀀스 크기보다 크다면, 해당 R*-트리를 사용할 수 없다[4]. 따라서 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.

일반적으로 R*-트리 구성을 위한 윈도우 크기는 해당 응용에서 사용 가능한 최소의 질의 시퀀스의 크기 minQLen(FRM의 경우) 혹은 ⌊(minQLen+1)/2⌋(Dual-Match의 경우)가 된다. 그러나 실제 응용에서 사용되는 질의 시퀀스의 크기는 매우 다양하므로 이와 같이 윈도우의 크기와 질의 시퀀스 크기의 차가 큰 경우에는 전체 수행 시간이 매우 커진다.

본 논문에서는 다양한 실험을 통하여 윈도우 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.

3.2 사전 실험 설정

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

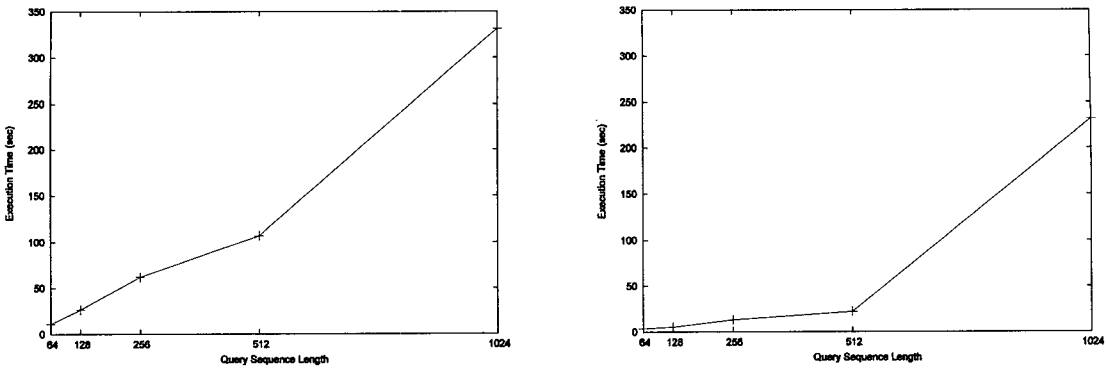
사전 실험은 두 가지로 구분하여 실행하였다. 첫번째 실험은 고정된 크기의 윈도우로 구성된 하나의 인덱스를 이용하여 여러 가지 길이의 질의 시퀀스들에 대한 서브시퀀스 매칭을 수행하여 질의 시퀀스의 길이에 따른 성능 변화를 살펴본다. 첫번째 실험에서 사용된 윈도우 크기는 w = 64이며, 질의 시퀀스의 길이는 Len(Q) = 64, 128, 256, 512, 1024이다. 두번째 실험은 첫번째 실험과 반대로 고정된 길이의 질의 시퀀스에 대하여 다양한 크기의 윈도우들로 구성된 각각의 인덱스를 이용하여 서브시퀀스 매칭을 수행함으로써 윈도우의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

성능 지수로는 검색에 소요된 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

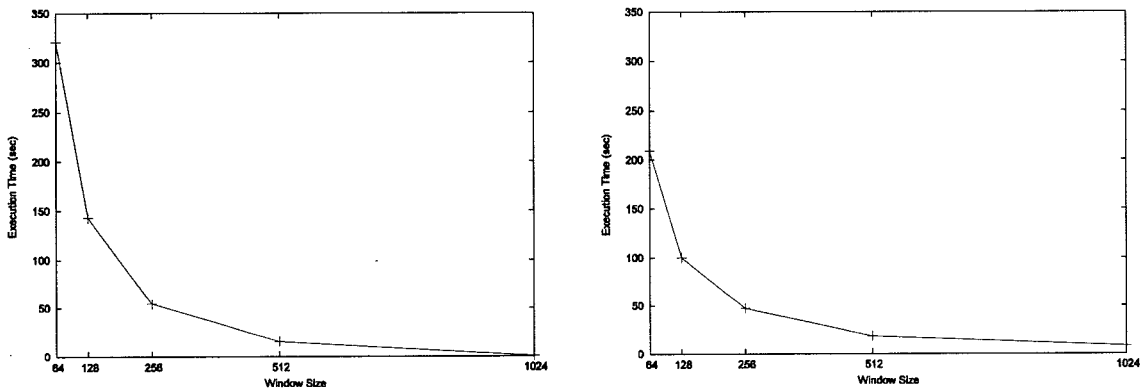
그림 1(a)와 (b)는 각각 FRM과 Dual-Match에 대한 첫번째 사전 실험 결과를 보인 것이다. 그림 3에서 가로축은 질의 시퀀스의 길이를 나타내고, 세로 축은 실행 시간을 나타낸다. 그림에 나타난 값들의 단위는 초(second)이다.

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



(a) FRM (b) Dual-Match

그림 1 질의 시퀀스의 길이에 따른 실행 시간의 변화



(a) FRM (b) Dual-Match

그림 2 윈도우 크기에 따른 실행 기간의 변화

검색 결과 후보 집합에 포함되는 후보 서브시퀀스의 개수가 증가하기 때문이다.

그림 2(a)와 (b)는 각각 FRM과 Dual-Match에 대한 두번째 사전 실험 결과를 보인 것이다. 가로 축은 인덱스를 구성한 윈도우의 크기를 나타내고 세로축은 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.

사전 실험의 결과를 요약하면, 질의 시퀀스의 크기와 윈도우의 크기의 차이가 커짐에 따라서 서브시퀀스 매칭의 성능은 급격하게 저하되는 것으로 나타났다. 따라서 기존 기법에서와 같이 하나의 윈도우 크기를 대상으로 생성된 단 하나의 인덱스만을 이용하는 경우, 실제 응용에서 만족할만한 성능을 낼 수 없다는 것을 알 수 있다.

4. 제안하는 기법

본 장에서는 서브시퀀스 매칭 수행 시에 윈도우 크기 효과에 의한 성능 저하를 극복하기 위한 방법으로 인덱스 보간법(index interpolation)[18]을 이용하는 새로운 기법을 제안한다. 인덱스 보간법은 먼저 서로 다른 크기의 윈도우들을 대상으로 다수의 인덱스들을 구축한 후, 서브시퀀스 매칭시 주어진 질의 시퀀스의 길이에 가장 적합한 하나의 인덱스를 인덱스 검색 단계에서 사용하는 기법이다.

일반적으로, 구성된 인덱스들의 수가 늘어날수록 서브시퀀스 매칭의 처리 성능은 향상된다. 하지만, 늘어난 인덱스를 관리하기 위한 비용이 증가하는 문제가 발생된다. 인덱스 관리 비용은 인덱스를 저장하기 위한 물리적 공간뿐만 아니라, 데이타가 삽입, 삭제, 갱신될 때마다 해당 인덱스들을 변경하는 비용까지 포함한다. 따라

서, 인덱스 보간법에서는 최적의 검색 성능을 지원할 수 있는 가능한 한 적은 수의 인덱스들을 선택하여 구성하는 것이 대단히 중요하다.

본 장에서는 물리적 데이타베이스 설계 관점에서 효율적인 다중 인덱스들의 구성을 위한 최적의 윈도우 크기 선정 방안에 관하여 논의한다. 제4.1절에서는 여러 개의 윈도우 크기가 주어졌을 때 주어진 질의 시퀀스의 길이에 가장 적합한 하나의 윈도우를 선택하는 방법에 대해서 설명하고 제4.2절에서는 여러 개의 질의 시퀀스의 길이와 빈도수가 주어졌을 때 서브시퀀스 매칭의 처리 비용 공식을 제안하고, 이 공식을 기반으로 최적의 윈도우 크기들을 선정하는 알고리즘을 제안한다.

4.1 윈도우 크기의 선택

이후의 논의 전개를 위하여 몇 가지 중요한 기호를 정리한다. 먼저, 응용에서 사용될 질의 시퀀스의 길이와 빈도수를 각각 $l_i (i \geq 1)$와 f_i로 표기한다. 또한, 이러한 질의 시퀀스 길이의 리스트와 빈도수 리스트를 각각 $L=<l_1, l_2, ..., l_n>$과 $F=<f_1, f_2, ..., f_n>$로 표기한다. 여기서 n은 질의 시퀀스 길이의 종류를 의미하며, $l_1 < l_2 < \cdots < l_n$의 관계가 성립한다고 가정한다. 인덱스 보간법을 이용한 인덱싱에서 채택된 윈도우의 크기를 $w_j (j \geq 1)$라고 표기한다. 응용에서 인덱스 보간법을 이용하여 구성한 m개의 인덱스에 사용된 윈도우 크기 리스트를 $W=<w_1, w_2, ..., w_m>$로 정의하며, 이 때에 $w_1 < w_2 < \cdots < w_m$의 관계가 성립한다고 가정한다.

서브시퀀스 매칭은 윈도우 크기 리스트 W내의 하나의 윈도우 크기를 선택하여 그것과 대응되는 인덱스를 이용하여 처리 된다[18]. 본 장에서는 길이가 l_k인 질의 시퀀스가 주어졌을 때 윈도우 크기 리스트에서 가장 적합한 윈도우 크기 $w_{\max}(l_k)$를 계산하는 방법을 제시한다. 먼저 $w_{\max}(l_k)$를 계산하는 공식을 제시하고 이 공식에 의해 선택된 $w_{\max}(l_k)$가 최적의 윈도우 크기임을 보인다.

$$w_{\max}(l_k) = \max\{w_i | w_i \leq l_k (1 \leq i \leq m)\} \text{ (FRM) 또는}$$
$$w_{\max}(l_k) = \max\{w_i | w_i \leq \lfloor (l_k+1)/2 \rfloor (1 \leq i \leq m)\}$$
Dual-Match) (1)

공식 (1)에 의하여 구한 $w_{\max}(l_k)$가 최적임을 보이는 과정은 다음과 같다. 먼저 위의 공식에 의하여 구해진 윈도우 크기 $w_{\max}(l_k)$를 사용하면 서브시퀀스 매칭에서 착오기각이 발생하지 않음을 보이고 $w_{\max}(l_k)$가 착오기각이 발생하지 않는 윈도우의 크기 중에서 가장 큰 윈도우 크기임을 이용하여 최적의 윈도우 크기임을 보인다.

보조 정리 1. 공식 (1)에 의해 선택된 윈도우 크기를 이용하여 수행한 인덱스 보간법 기반 서브시퀀스 매칭에서는 착오 기각이 발생하지 않는다.

증명. 참고문헌 [4,5]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 각 질의 시퀀스의 길이 l_k에 대하여 $w \leq l_k$ (FRM의 경우), 또는 $w \leq \lfloor (l_k+1)/2 \rfloor$ (Dual-Match의 경우)인 관계가 성립해야 한다는 것이다.

인덱스 보간법 기반 서브시퀀스 매칭에서 임의의 질의 시퀀스 l_k는 공식 (1)에 의해 선택된 인덱스를 이용하여 처리되므로 결국 위의 조건을 항상 만족하게 된다. 따라서 인덱스 보간법을 기반으로 하는 서브시퀀스 매칭에서는 모든 입력 가능한 질의 시퀀스에 대하여 공식 (1)에 의해 선택된 인덱스를 이용하여 서브시퀀스 매칭을 수행할 때, 착오 기각은 발생하지 않는다. □

위의 공식 (1)에 의해 선택된 윈도우 크기 $w_{\max}(l_k)$와 대응되는 인덱스를 이용하는 경우, 다른 윈도우 크기 $w' (\neq w_{\max}(l_k))$에 대한 인덱스를 이용하는 경우보다 더 좋은 성능을 기대할 수 있다. 그 이유는 다음과 같이 설명될 수 있다. 만약 w'이 $w_{\max}(l_k)$보다 작다면 $w' < w_{\max}(l_k) \leq l_k$ (FRM의 경우) 또는 $w' < w_{\max}(l_k) \leq \lfloor (l_k+1)/2 \rfloor$ (Dual-Match의 경우)인 관계가 성립한다. 윈도우 크기 효과에 의하여 w'크기의 윈도우로 구축한 인덱스를 이용한 서브시퀀스 매칭의 성능은 $w_{\max}(l_k)$크기의 윈도우로 구축한 인덱스를 이용한 경우와 비교하여 그 성능이 떨어지게 된다.

반면에, 만약 w'이 $w_{\max}(l_k)$보다 크다면 $w_{\max}(l_k)$의 정의에 의하여 $w' > l_k$ (FRM의 경우) 또는 $w' > \lfloor (l_k+1)/2 \rfloor$ (Dual-Match의 경우)임을 의미한다. 따라서 이 경우에는 FRM과 Dual-Match에서 모두 w'크기의 윈도우들을 대상으로 구축된 인덱스를 이용하여 서브시퀀스 매칭을 수행할 수 없으므로 반드시 순차 검색을 수행하여야 한다. 순차 검색은 인덱스를 이용한 검색에 비하여 성능이 크게 떨어지므로 전체 서브시퀀스 매칭의 성능을 저하시키게 된다.

4.2 최적의 다중 인덱스 구성 방안

인덱스 보간법은 둘 이상의 인덱스를 사전에 구성해 두고, 주어진 질의 시퀀스에 대하여 가장 적합한 인덱스를 선택하여 서브시퀀스 매칭을 수행한다는 기본적인 원칙일 뿐 어떤 크기의 윈도우들을 대상으로 인덱스들을 구성할 것인가는 지정하지 않는다[18]. 본 절에서는 물리적 데이타베이스 설계 기법을 이용한 최적의 서브시퀀스 매칭을 지원하는 다중 인덱스 구성 방안에 대하

여 논의한다. 물리적 데이타베이스 설계는 주어진 질의 집합을 대상으로 최적의 질의 처리를 제공하는 물리적 파일 구조 및 액세스 구조를 결정하는 과정이다[25]. 제안된 기법은 응용에서 사용될 질의 시퀀스들에 대한 정보로서 <길이, 빈도>의 쌍과 응용에서 구성할 인덱스의 개수 m을 입력으로 받아 서브시퀀스 매칭을 최적으로 처리할 수 있는 인덱스들을 구성할 수 있도록 윈도우 크기 $w_1, w_2, \cdots, w_m$를 결정한다. 먼저, 전체 서브시퀀스 매칭 처리를 위한 비용 공식을 도출하고, 그 공식에 기반하여 전체 질의 시퀀스들을 대상으로 최적의 서브시퀀스 매칭 수행을 위한 윈도우 크기를 선정하는 알고리즘을 제시한다.

주어진 질의 시퀀스의 길이 l_k와 응용에서의 사용 빈도수 f_k, 윈도우 크기 리스트 W를 이용하여 서브시퀀스 매칭을 수행하였을 때의 비용을 $C(l_k, f_k, W)$라고 표기한다. 본 연구에서는 이 비용을 아래와 같이 추정한다. 이는 제3장의 사전 실험 결과에서 서브시퀀스 매칭의 성능이 질의 시퀀스의 길이에 비례하며 윈도우 크기에 반비례한다는 관찰을 토대로 한 것이다. 즉, 이 비용 공식은 처리 비용에 영향을 미칠 수 있는 다른 요소들을 배제하고, 본 연구의 관심의 대상인 질의 시퀀스의 길이와 윈도우 크기만을 고려한 것이다.

$$C(l_k, f_k, W) = f_k \cdot \frac{l_k}{w_{\max}(k)} \quad (2)$$

위의 공식을 확장하여 응용에서 사용될 모든 질의 시퀀스들을 대상으로 서브시퀀스 매칭을 수행하기 위한 전체 비용 T를 추정하면 아래의 공식 (3)과 같다. 여기서, n은 사용될 질의 시퀀스 길이 종류의 수이다.

$$T = C(L, F, W) = \sum_{k=1}^{n} \left(f_k \cdot \frac{l_k}{w_{\max}(l_k)} \right) \quad (3)$$

추가적으로 질의 시퀀스 길이 리스트 L의 부분 리스트 $<l_i, \ldots, l_j>$를 $L[i..j]$라고 표기할 때, $L[i..j]$의 실행 비용 $C(L[i..j], F[i..j], W)$은 공식 (3)으로부터 다음과 같이 유도할 수 있다.

$$C(L[i..j], F[i..j], W) = \sum_{k=i}^{j} \left(f_k \cdot \frac{l_k}{w_{\max}(k)} \right) \quad (4)$$

본 논문에서 해결하고자 하는 문제는 공식 (3)의 비용 T를 최소로 하는 윈도우 크기 리스트 $W=<w_1, w_2, \ldots, w_m>$를 효과적으로 구하는 것이다. 여기서, 이러한 W를 최적의 윈도우 크기 리스트 $W_{optimal}$이라 정의한다. m개로 구성된 모든 가능한 윈도우 크기 리스트에 대하여 T를 구하면, 최소의 T값을 갖는 W를 찾아낼 수 있다. 그러나 모든 가능한 윈도우 크기 리스트의 수는 ${}_{l_n}C_m$개 이므로 이러한 방식의 알고리즘은 $O((l_n)^m)$의 지수적 시간 복잡도를 가지게 된다. 본 논문에서는 $O(mn^2)$의 시간 복잡도를 갖는 알고리즘을 제시한다. 먼저 ${}_{l_n}C_m$개의 모든 가능한 윈도우 크기 리스트를 고려할 필요 없이 ${}_nC_m$개의 윈도우 크기 리스트만 고려하면 최적의 윈도우 크기 리스트를 찾을 수 있음을 보이고 ${}_nC_m$개의 윈도우 크기 리스트에서 $O(mn^2)$ 시간에 최적의 윈도우 크기 리스트를 찾는 알고리즘을 소개한다.

다음의 보조정리 2는 최적 윈도우 크기 리스트 $W_{optimal}$내의 각 윈도우 크기는 반드시 질의 시퀀스 길이 리스트 L내의 한 요소 l_j와 대응된다는 사실을 소개한다. 이 사실은 최적의 윈도우 크기 리스트는 n개의 질의 시퀀스의 길이 중 m개를 선택하여 만든 윈도우 리스트 중의 하나임을 의미하며 이는 곧 우리가 최적의 윈도우 크기 리스트를 찾기 위해서 ${}_nC_m$개의 윈도우 크기 리스트만을 고려하면 된다는 뜻이 된다.

보조정리 2. 윈도우 크기 리스트 $W_{optimal} = <w_1, w_2, \ldots, w_m>$를 $L=<l_1, l_2, \ldots, l_n>$ 및 $F=<f_1, f_2, \ldots, f_n>$의 서브시퀀스들을 위한 최적의 윈도우 크기 리스트라 하면, $<w_1, w_2, \ldots, w_m> \geq <l_{g(1)}, l_{g(2)}, \ldots, l_{g(m)}>$이 성립한다. (여기서, $1 \leq g(1) < g(2) < \cdots g(m) \leq n$).

증명. 각각의 윈도우 크기 $w_i (1 \leq i \leq m)$는 어떤 $l_j (1 \leq j \leq n)$와 항상 일치함을 귀류법을 이용하여 증명한다. 만일, w_i가 어떠한 l_j와도 일치하지 않는 윈도우 크기라고 가정하면, $l_{a-1} < w_i < l_a$를 만족하는 연속적인 질의 시퀀스의 길이 l_{a-1}과 l_a가 존재한다. 그러나 이 경우, 윈도우 크기 리스트 $<w_1, \ldots, w_i, \ldots, w_m>$를 이용한 길이와 빈도가 각각 L, F의 서브시퀀스들의 처리 비용은 윈도우 크기 $<w_1, \ldots, l_a, \ldots, w_m>$을 이용한 처리 비용보다 항상 크게 된다. 이는 $<w_1, w_2, \ldots, w_m>$이 최적의 윈도우 크기 리스트라는 가정에 모순이 된다. 그러므로 각각의 윈도우 크기 w_i는 L내의 어떤 l_j와 일치한다. □

이제 ${}_nC_m$ 개의 윈도우 크기 리스트에서 $O(mn^2)$ 시간에 최적의 윈도우 크기 리스트를 찾는 알고리즘을 소개한다. $L=<l_1, l_2, \ldots, l_n>$, $F=<f_1, f_2, \ldots, f_n>$이 주어졌을 때 최적의 윈도우 크기 리스트 $W_{optimal}$는 최소 실행 비용 $C(L, F, W_{optimal})$를 구하는 과정의 부산물로서 쉽게 얻을 수 있으므로, $C(L, F, W_{optimal})$를 구하는 과정에 대해서만 설명하기로 한다. 모든 ${}_nC_m$ 개의 윈도우 크기 리스트에 대하여 실행 비용을 구하고 그 중의 최소값을 구하면 최소 실행 비용을 구할 수 있다. 그러나 이러한 방식의 알고리즘은 $O(n^m)$의 시간 복잡도를 가지게 된다. 본 논문에서 제시하는 $O(mn^2)$ 시간 복잡도

알고리즘의 기본 아이디어는 다이나믹 프로그래밍 기법을 이용하여 부분 질의 시퀀스 길이 리스트들에 대해서 최소 실행 비용을 먼저 구하고 그 결과를 이용하여 전체 질의 시퀀스 길이 리스트들에 대해서 최소 실행 비용을 구하는 것이다.

$C(L, F, W_{optimal})$를 구하는 과정은 크게 두 단계로 구성된다. 단계 1에서는 $n \times n$ 크기의 NC테이블 내의 각 요소 $NC(i,j)(1 \le i \le j \le n)$에 $C(L[i..j], F[i..j], < l_i >)$값을 저장한다. $C(L[i..j], F[i..j], < l_i >)$는 윈도우 크기 리스트 $< l_i >$를 이용하여 $< l_i, ..., l_j >$의 부분리스트를 처리하는 비용이다. 단계 2에서는 NC테이블을 이용하여 최소 실행 비용인 $C(L, F, W_{optimal})$를 계산한다.

단계 1. NC테이블의 계산 : 각각의 i, $j(1 \le i \le j \le n)$에 대하여 $C(L[i..j], F[i..j], < l_i >)$를 계산한 결과를 $NC(i,j)$에 저장한다. 식 (4)에 의해서 $NC(i,j) = \sum_{k=i}^{j} \left(f_k \cdot \frac{l_k}{l_i} \right)$ 이므로 모든 i, $j(1 \le i \le j \le n)$에 대하여 다음의 두 식 $NC(i,i) = f_i$과 $NC(i,j) = NC(i,j-1) + f_j \cdot \frac{l_j}{l_i}$ 이 성립한다. 이 두 식에 의하면, $NC(i,i)$는 상수 시간에 직접 계산 할 수 있고, $NC(i,j)$는 $NC(i,j-1)$로부터 상수 시간에 계산할 수 있다. 그러므로 다음의 보조정리를 얻을 수 있다.

보조정리 3. 모든 $NC(i,j)(1 \le i \le j \le n)$를 구하는 작업의 시간 복잡도는 $O(n^2)$이다.

단계 2. 최소 비용 $C(L, F, W_{optimal})$의 계산 : $C'(i,j)(1 \le i \le n, 1 \le j \le m)$을 가장 작은 윈도우 크기 w_1이 l_i이며, j개의 윈도우 크기들을 포함하는 리스트를 이용하여 부분 리스트 $(l_i, ..., l_n)$를 처리하는 최소 비용이라고 하자. 그렇다면 $C(L, F, W_{optimal}) = C'(1,m)$이 성립한다. 우리는 $C'(1,m)$을 동적 프로그래밍 기법을 이용하여 계산한다. 먼저 $C'(i,j)(1 \le i \le n, 1 \le j \le m)$에 대하여 다음의 순환 구조(recurrence)가 성립됨을 보인다.

보조정리 4.

$$C'(i,j) = \min_{k=i+1}^{n-j+2} NC(i,k-1) + C'(k,j-1)$$

증명. $C'(i,j)$의 정의에 의하여, 가장 작은 윈도우 크기 w_1은 l_i이다. 두 번째로 작은 윈도우 크기 w_2를 고려해 보자. w_2는 보조정리 2에 의하여 질의 시퀀스의 길이 $l_{i+1}, ..., l_{n-i+1}$중의 하나의 값을 갖는다. 만일 w_2가 l_k이면, 모든 $i+1 \le k' \le n-i+1$에 대하여 $C'(i,j) = NC(i,k-1) + C'(k,j-1) \le NC(i,k'-1) + C'(k',j-1)$를 만족한다. 그러므로 위의 순환 구조가 성립한다. □

보조정리 4에 의하면 $C'(i,j)$은 NC테이블과 $C'(k,j-1)$ $(1 \le k \le n-i+1)$ 값들을 이용하여 $O(n)$ 시간에 계산할 수 있다. $C'(1,m)$을 계산하기 위해서는 최대 mn개의 $C'(i,j)$값들을 계산하게 되므로 $C'(1,m)$의 계산 복잡도는 $O(mn^2)$이며, 이 결과 다음의 보조정리를 얻을 수 있다.

보조정리 5. 최소 비용 $C(L, F, W_{optimal})$의 계산 복잡도는 $O(mn^2)$이다.

그림 3은 이와 같은 개념을 이용한 $C'(1,m)$의 계산 과정을 의사 코드로 나타낸 것이다.

```
1 : for i 1 to n do
2 :    NC[i][i] := fi
3 :    for j := i+1 to n do
4 :       NC[i][j] := NC[i][j-1] + fi(lj/li)

5 : for i := 1 to n do
6 :    C[i][1] := NC[i][n]
7 :    for j := 2 to n-i do
8 :       C[i][j] := ∞
9 :    for j := n-i+1 to m do
10 :      C[i][j] := 0

11 : for i := n-2 downto 1 do
12 :    for j := 2 to min{m,n-i} do
13 :       for k := i+1 to n-j+2 do
14 :          temp := NC[i][k-1]+C[k][j-1]
15 :          if(temp<C[i][j]) C[i][j] := temp
```

그림 3 C′(1,m) 계산 과정의 의사 코드

각 라인별 설명은 다음과 같다. 1~4 라인 : NC테이블의 각 요소 값을 계산한다. 5~10 라인 : C' 테이블의 일부 요소들의 값을 초기화 한다. 11~15 라인 : C'테이블의 모든 요소의 값을 계산한다.

5. 성능 평가

본 장에서는 실험에 의한 성능 분석을 통하여 제안하는 기법의 우수성을 규명한다. 먼저, 제5.1절에서는 성능 평가를 위한 실험 환경을 설명하고, 제5.2절에서는 기존 기법과의 비교 실험을 통한 제안된 기법의 성능 개선 효과를 제시한다.

5.1 실험 환경

본 연구에서는 성능 분석을 위하여 실제 데이타베이스 K_Stock_Data와 합성 데이타 Syn_Data를 사용하였다. K_Stock_Data는 사전 실험에서 사용하였던 한국의 실제 주식 데이타로서 길이가 1,024인 620개의 데이타 시퀀스로 구성된다. 합성 데이타 Syn_Data내의 각 시퀀스 S = $<s_1, s_2, ..., s_n>$는 다음과 같은 랜덤 워크

표 1 그룹 내에 포함된 질의 시퀀스의 개수

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

(random walk) 형태를 가진다[1].

$$S_i = S_{i-1} + Z_i$$

여기서 z_i는 구간 [-0.1, 0.1] 사이에서 균일한 분포를 취하는 랜덤 변수이며, 시퀀스의 첫 요소 값 s_1은 구간 [1, 10] 사이의 임의의 값을 취하도록 하였다. 다양한 환경에 대한 실험을 위하여 각각 2,000개, 3,000개, 4,000개, 5,000개의 길이가 1,024인 데이타 시퀀스들로 구성된 다섯 가지 Syn_Data들과 길이가 각각 2,000, 3,000, 4,000, 5,000인 1,000개의 데이타 시퀀스들로 구성된 다섯 가지 Syn_Data들을 생성하였다.[3)]

질의 시퀀스의 길이는 64부터 1024까지의 범위내에서 32의 배수의 길이를 가지며, 같은 길이를 갖는 질의 시퀀스를 같은 그룹으로 구성하였다. 따라서 전체 질의 시퀀스 그룹은 31이 된다. 본 실험에서 동일한 수의 질의 시퀀스들을 포함하는 그룹의 개수와 그 그룹에 포함된 질의 시퀀스의 개수는 실제 응용의 특성을 반영하여 아래의 표 1과 같이 설정하였다. 4개의 그룹내의 질의 시퀀스들이 빈번하게 사용되고, 16개 그룹내의 질의 시퀀스들은 사용이 빈번하지 않음을 의미한다. 성능 평가지수로서 전체 그룹내의 총 216개의 질의 시퀀스를 대상으로 서브시퀀스 매칭을 수행하는 데에 걸린 실행 시간의 평균을 사용하였다. 각 질의 시퀀스에 대한 허용치 ε은 서브시퀀스 매칭의 결과로 20 개의 서브시퀀스를 반환하도록 조정하였다.

실험을 위한 환경으로는 2.8 GHz Pentium 4 프로세서에 512 MB의 메모리를 장착한 PC와 MS Windows 2000 운영 체제를 사용하였다. 본 실험에서는 일정한 실험 결과를 얻기 위하여 운영 체제에서 제공하는 버퍼링(buffering) 기능을 사용하지 않고 바로 디스크를 액세스하는 시스템 I/O 함수를 이용하였다. 또한, 다차원 인덱스로는 1KB 크기의 페이지를 갖는 R*-트리[17]를 사용하였다. 인덱싱을 위한 저차원 변환은 DFT를 통하여 수행하였으며, 특성 추출 계수 f=6으로 하였다. 서브 시퀀스 매칭을 위한 기법으로는 FRM에 비하여 더 우수한 성능을 보이는 것으로 규명된 바 있는[5] Dual-Match를 이용하였다.

3) 참고문헌 [5]의 성능 평가에서는 매우 긴 하나의 데이타 시퀀스로 구성된 데이타베이스를 사용하였다. 그러나 이 경우, 각 데이타 시퀀스를 액세스의 단위로 사용하는 실제 시계열 데이타베이스의 특성을 올바르게 반영하지 못하는 상황이 발생한다. 따라서 본 연구에서는 성능 평가를 위하여 실제 환경과 동일하게 다양한 길이를 가지는 다수의 데이타 시퀀스들로 구성되는 데이타베이스를 사용하였다.

본 실험에서의 성능 평가의 대상으로 선정한 서브시퀀스 매칭 기법은 단일 인덱스만을 이용하는 기존의 기법(A), 최소 질의 시퀀스 길이와 최대 시퀀스 길이 범위내에서 균등한 간격으로 윈도우 크기를 설정하여 구성한 다중 인덱스들을 이용하는 기법(B), 본 논문에서 제안한 기법으로 생성한 다중 인덱스들을 이용하는 기법(C) 등 세가지이다.

5.2 실험 결과

본 논문에서는 세 가지 실험을 수행한다. 실험 1에서는 실제 데이타를 이용하여 인덱스의 개수를 변경하며 세가지 기법들간의 성능을 비교한다. 실험 2에서는 합성 데이타를 이용하여 데이타 시퀀스의 개수를 변경하면서 세가지 기법들간의 성능을 비교한다. 실험 3에서는 합성 데이타를 이용하여 데이타 시퀀스의 길이를 변경하며 세가지 기법들간의 성능을 비교한다.

실험 1에서 기법(A)는 w=32인 윈도우를 대상으로 하나의 인덱스만을 사용하였으며, 기법 (B)에서는 크기가 일정한 간격으로 설정된 각각 3, 4, 5가지의 윈도우들을 대상으로하는 다중 인덱스를 사용하였고, 기법 (C)에서는 본 논문에서 제안한 알고리즘으로 선택된 3, 4, 5가지의 윈도우들을 대상으로하는 다중 인덱스를 사용하였다.

그림 4는 실험 1의 결과를 보인 것이다. 가로 축은 인덱스의 개수, 세로 축은 실행 시간의 평균을 초 단위로 나타낸다.

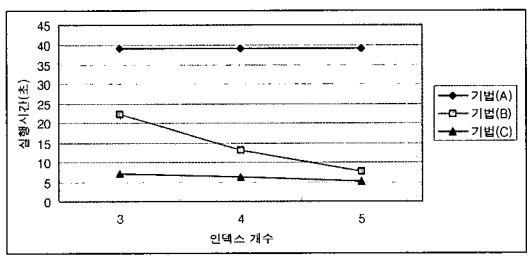


그림 4 인덱스의 개수에 의한 성능 비교

실험 결과, 하나의 인덱스를 사용하는 기법(A)에 비하여 일정 간격의 다중 인덱스를 이용한 기법(B)가 좋은 성능을 보였고, 기법(B)에 비하여 제안된 기법에 의하여 선정된 다중 인덱스를 이용한 기법(C)가 더 좋은 성능을 보였다. 그림 4에서 5개의 인덱스를 이용한 경우, 기법(C)는 기법(A)에 비하여 약 7.8배, 기법(B)에

비하여 약 1.5배 성능이 향상되었다. 또한, 3개의 인덱스를 이용한 경우, 기법(C)는 기법(A)에 비하여 약 5.6배, 기법(B)에 비하여 약 3.2배 성능이 향상되었다.

실험 2에서는 1024의 길이를 갖는 합성 데이타를 2000, 3000, 4000, 5000 개로 개수를 증가시키면서, 세 가지 기법의 성능의 변화를 관찰하였다. 기법(B)와 기법(C)에서는 각각 4개의 다중 인덱스를 사용하도록 설정하였다.

그림 5는 실험 2의 결과를 보인 것이다. 가로 축은 데이타 시퀀스의 개수, 세로 축은 실행 시간의 평균을 초 단위로 나타낸 것이다.

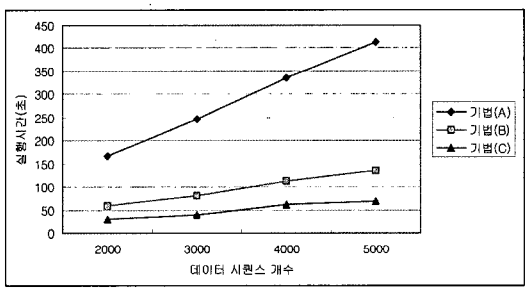


그림 5 데이타 시퀀스 개수에 의한 성능 비교

데이타 시퀀스의 개수가 증가하더라도 기법(A)에 비하여 기법(B)가 좋은 성능을 보였고, 기법(B)에 비하여 제안된 기법에 의하여 정해진 다중 인덱스를 이용한 기법(C)가 더 나은 성능을 보였다. 기법(C)는 기법(A)에 비하여 약 5.7배에서 6.1배까지, 기법(B)에 비하여 약 1.9배에서 2.1배까지 성능이 향상되었다.

실험 3에서는 2000, 3000, 4000, 5000의 길이를 갖는 1000개의 합성 데이타를 대상으로 세가지 기법의 성능의 변화를 관찰하였다. 실험 2에서와 마찬가지로, 기법(B)와 기법(C)에서는 각각 4개의 다중 인덱스를 사용하도록 설정하였다.

그림 6은 실험 3의 결과를 보인 것이다. 가로 축은 데이타 시퀀스의 개수, 세로 축은 실행 시간의 평균을 초 단위로 나타낸 것이다.

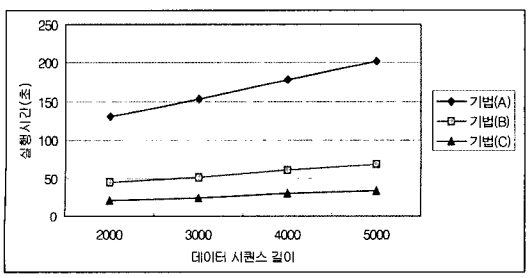


그림 6 데이타 시퀀스 길이에 의한 성능 비교

실험 결과는 실험 2와 그 경향이 매우 유사하게 나타났다. 기법(C)는 데이타 시퀀스의 길이에 상관 없이 기법(A)와 비교하여 약 6.2배, 기법(B)에 비하여 약 2배의 성능이 개선되는 것으로 나타났다.

전체 실험 결과들을 요약하면 다음과 같다. 단 하나의 인덱스만을 이용하는 기존의 기법과 비교하여, 다중 인덱스를 사용하는 방식을 채택함으로써 큰 성능 개선 효과를 얻을 수 있었다. 또한, 사용되는 질의 시퀀스 길이의 분포를 고려하지 않은 일정 간격의 단순한 다중 인덱스를 이용하는 기법(B)와 비교해서도 제안된 기법(C)는 상당한 실행 시간의 단축 효과를 보이는 것으로 나타났다.

6. 결 론

본 논문에서는 기존의 기법에서 윈도우 크기 효과에 의하여 검색 성능이 저하되는 문제점을 해결하고자 인덱스 보간법 [18]에 기반한 새로운 서브시퀀스 매칭 기법을 제안하였다. 인덱스 보간법이란 하나 이상의 인덱스를 구축하고, 주어진 질의 시퀀스의 길이에 따라 하나의 적절한 인덱스를 선택하여 검색을 수행함으로써 서브시퀀스 매칭 성능을 향상시키는 기법이다.

본 논문의 주요 공헌은 다음과 같다.

- 단 하나의 인덱스만을 이용하는 기존의 서브시퀀스 매칭 기법은 실제 응용에서 만족할만한 성능을 낼 수 없음을 실험을 통하여 정량적으로 규명하고, 이러한 문제의 해결을 위하여 인덱스 보간법의 응용이 필요함을 보였다.
- 데이타베이스에서 수행될 예정인 질의 시퀀스들의 집합이 주어지고, 인덱스 구성의 기반이 되는 윈도우들의 크기들이 결정되었을 때, 전체 서브시퀀스 매칭들을 수행하는 데에 소요되는 비용을 예측할 수 있는 공식을 산출하였다.
- 수행 비용 공식을 이용하여 서브시퀀스 매칭의 성능을 극대화 할 수 있도록 최적의 윈도우 크기들을 결정하는 효율적인 알고리즘을 제안하였다.
- 제안된 알고리즘의 최적성과 효율성을 이론적으로 규명하였다.
- 제안된 기법이 단순한 기존의 기법과 비교하여 상당한 성능 개선 효과가 있음을 다양한 실험을 통해서 검증하였다.

실제 주식 데이타를 사용한 실험 결과, 제안된 기법은 단일 인덱스를 사용한 경우와 비교하여 약 7.8배, 일정 구간 인덱스를 사용한 경우와 비교하여 1.5배의 성능 향상을 보였다. 또한, 대량의 합성 데이타를 이용한 실험 결과, 제안된 기법은 단일 인덱스를 사용한 경우와 비교하여 데이타 시퀀스의 개수 및 길이에 상관 없이 약 6

배의 성능 향상을 보였으며, 일정 간격의 인덱스를 사용한 경우와 비교하여 약 2배의 성능 향상을 보였다.

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] R. Agrawal et al., "Fast Similarity Seasch 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.

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

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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, 1999.

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

[10] 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*, CP, pp. 137-153, Sept. 1995.

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

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

[13] D. J. Berndt and J. Clifford, "Finding Patterns in Time Series: A Dynamic Programming Approach," *Advances in Knowledge Discovery and Data Mining*, pp. 229-248, 1996.

[14] B. K. Yi, H. V. Jagadish, and C. Faloutsos, "Efficient Retrieval of Similar Time Sequences Under Time Warping," In *Proc. Int'l. Conf. on Data Engineering*, IEEE ICDE, pp. 201-208, 1998.

[15] S. H. Park et al., "Efficient Searches for Similar Subsequences of Difference Lengths in Sequence Databases," In *Proc. Int'l. Conf. on Data Engineering*, IEEE ICDE, pp. 23-32, 2000.

[16] 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. Int'l. Conf. on Data Engineering*, IEEE ICDE, pp. 607-614, 2001.

[17] 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.

[18] W. K. Loh, S. W. Kim, and K. Y. Whang, "Index Interpolation: A Subsequence Matching Algorithm Supporting Moving Average Transform of Arbitrary Order in Time-Series Databases," *IEICE Trans. on Information and Systems*, Vol. E84-D, No. 1, pp. 76-86, 2001.

[19] 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.

[20] 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.

[21] G. Das, D. Gunopulos, and H. Mannila, "Finding Similar Time Series," In *Proc. European Symp. on Principles of Data Mining and Knowledge Discovery*, PKDD, pp. 88-100, 1997.

[22] 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.

임 승 환

2003년 2월 한양대학교 전자전기컴퓨터공학부 졸업(학사). 2005년 2월 한양대학교 정보통신대학원 졸업(석사). 관심분야는 사회연결망 분석

김 상 욱

1989년 2월 서울대학교 컴퓨터공학과 졸업(학사). 1991년 2월 한국과학기술원 전산학과 졸업(석사). 1994년 2월 한국과학기술원 전산학과 졸업(박사). 1991년 7월~8월 미국 Stanford University, Computer Science Department 방문 연구원. 1994년 2월~1995년 2월 KAIST 정보전자연구소 전문 연구원. 1999년 8월~2000년 8월 미국 IBM T.J. Watson Research Center Post-Doc. 1995년 3월~2003년 2월 강원대학교 컴퓨터정보통신공학부 부교수. 2003년 3월~현재 한양대학교 정보통신대학 정보통신학부 교수. 관심분야는 데이타베이스 시스템, 저장 시스템, 데이타 마이닝, 멀티미디어 정보 검색, 공간 데이타베이스/GIS, 주기억 장치 데이타베이스, 트랜잭션 관리

박 희 진

1994년 2월 서울대학교 컴퓨터공학과 졸업(학사). 1996년 2월 서울대학교 컴퓨터공학과 졸업(석사). 2001년 2월 서울대학교 컴퓨터공학과 졸업(박사). 2001년 3월~2003년 2월 서울대학교 컴퓨터연구소 전문연구원. 2003년 3월~2003년 8월 이화여자대학교 컴퓨터학과 BK연구교수. 2003년 9월~현재 한양대학교 정보통신대학 컴퓨터전공 조교수. 관심분야는 컴퓨터 알고리즘, 생물정보학, 암호학, 컴퓨터 보안