산업기술연구(강원대학교 산업기술연구소 논문집), 제21권 A호, 2001.
Journal of Industrial Technology, Kangwon Nat'l Univ.,Korea, No. 21 A, 2001.

타임 워핑을 지원하는 효율적인 서브시퀀스 매칭 기법

A Subsequence Matching Technique that Supports Time Warping Efficiently

박 상 현* 김 상 욱** 조 준 서* 이 헌 길**
Park, Sang-Hyun Kim, Sang-Wook Cho, June-Suh Lee, Hoen-Gil

Abstract

This paper discusses an index-based subsequence matching that supports time warping in large sequence databases. Time warping enables finding sequences with similar patterns even when they are of different lengths. In earlier work, we suggested an efficient method for whole matching under time warping. This method constructs a multidimensional index on a set of feature vectors, which are invariant to time warping, from data sequences. For filtering at feature space, it also applies a lower-bound function, which consistently underestimates the time warping distance as well as satisfies the triangular inequality.

In this paper, we incorporate the *prefix-querying approach* based on sliding windows into the earlier approach. For indexing, we extract a feature vector from every subsequence inside a sliding window and construct a multi-dimensional index using a feature vector as indexing attributes. For query processing, we perform a series of index searches using the feature vectors of *qualifying* query prefixes. Our approach provides effective and scalable subsequence matching even with a large volume of a database. We also prove that our approach does not incur false dismissal. To verify the superiority of our method, we perform extensive experiments. The results reveal that our method achieves significant speedup with real-world S&P 500 stock data and with very large synthetic data.

키워드 : *시퀀스 데이터베이스, 타임워핑 지원 서브시퀀스 검색*
Keywords : sequence database, subsequence matching with time warping

1. 서론

시퀀스 데이터베이스(sequence database)란 객체의 변화되는 값들의 연속으로 구성된 데이터 시퀀스(data sequence: 이후부터 간략히 시퀀스라 칭함)들의 집합이다[1]. 대표적인 예로는 주가 데이터, 환율 데이터, 기온 데이터, 제품 판매량 데이터, 기업 성장률 데이터 등이 있다[2][10]. 유사 검색(similarity search)이란 주어진 질의 시퀀스(query sequence)와 변화의 패턴이 유사한 시퀀스들을 시퀀스 데이터베이스로부터 찾아내는 연산이다[1][2][10]. 이러한 유사 검색은 데이터 마이닝(data mining) 및 데이터 웨어하우징(data warehousing) 분야에서 중요한 연산으로 사용된다

* IBM T.J. 왓슨 연구소
** 강원대학교 컴퓨터정보통신공학부
※ 본 논문은 정보통신부 주관 정보통신 우수시범학교 지원 사업과 강원대학교 BK21 지원 사업의 결과임

[7][23].

유사 검색은 전체 매칭(whole matching)과 서브시퀀스 매칭(subsequence matching)으로 구분된다[1]. 전체 매칭은 시퀀스들과 질의 시퀀스의 길이가 동일하다는 조건하에 수행되며, 질의 시퀀스와 유사한 시퀀스를 검색한다. 반면, 서브시퀀스 매칭은 이러한 조건이 불필요하며, 질의 시퀀스와 유사한 서브시퀀스를 포함하는 시퀀스를 검색한다.

유사 검색에 관한 기존의 많은 연구에서는 길이 n의 시퀀스를 n 차원 공간상의 한 점으로 간주한다. 또한, 두 시퀀스들간의 유사한 정도를 측정하기 위하여 두 점들간의 유클리드 거리(Euclidean distance)를 이용한다[1][8][10][12][23].

유클리드 거리만을 이용한 유사 검색을 통해서는 사용자가 원하는 시퀀스들을 검색하지 못하는 경우가 빈번하게 발생한다. 따라서 응용 분야에 적합한 유사 모델(similarity model)을 적절하게 정의할 수 있도록 변환(transform)을 지원하기도 한다. 초기의 연구인 참고 문헌 [1][10] 등에서는 변환을 지원하지 않았으나, 이후에는 스케일링(scaling)[2][8], 시프팅(shifting)[2][8], 정규화(normalization)[9][12][19], 이동 평균(moving average)[23][20], 타임 워핑(time warping)[4][27][21] 등의 다양한 변환을 지원하는 방법들이 제안되었다.

이들 중 타임 워핑은 시퀀스내의 각 요소 값을 임의의 수만큼 반복시키는 것을 허용하는 변환이다[27]. 예를 들어, 타임 워핑에 의하여 두 시퀀스 S = <20, 21, 21, 20, 20, 23, 23, 23> 와 Q = <20, 20, 21, 20, 23>를 동일한 시퀀스 <20, 20, 21, 21, 20, 20, 23, 23, 23>으로 변환시킬 수 있다. 타임 워핑 후의 두 시퀀스들 간의 거리를 타임 워핑 거리(time warping distance)라 정의한다. 유클리드 거리는 시퀀스들의 길이가 동일한 경우에만 유사한 정도를 측정할 수 있다. 따라서, 타임 워핑은 데이터베이스내의 시퀀스들의 길이가 서로 달라서 유클리드 거리를 이용하여 유사 정도를 직접 측정할 수 없는 경우에 매우 유용하다.

효율적인 유사 검색을 위하여 기존의 연구에서는 다차원 인덱스(multidimensional index)[3][5][24]를 사용한다[1][2][10]. 대부분의 인덱스들은 채택하는 거리 함수가 삼각형 부등식 성질(triangle inequality)[22]을 만족한다는 것을 전제로 한다. 만일, 이 성질을 만족하지 못하는 거리 함수를 이용하는 경우에는 유사 검색 시 착오 기각(false dismissal)이 발생된다[27]. 착오 기각이란 실제 질의 결과로 반환되어야 할 질의 시퀀스와 유사한 시퀀스를 올바르게 찾아내지 못하는 현상이다[1][10]. 참고 문헌 [27]에서는 타임 워핑 거리가 삼각형 부등식 성질을 만족하지 못함을 증명하고, 착오 기각을 허용하지 않는 응용에서 타임 워핑을 지원하는 유사 검색을 처리할 때에는 거리 함수 기반 인덱스를 사용할 수 없다고 주장한 바 있다.

참고 문헌 [16]에서는 인덱스를 기반으로 타임 워핑을 지원하는 효과적인 전체 매칭 기법을 제안한 바 있다. 이 기법은 데이터베이스 내의 각 시퀀스로부터 추출된 특징(feature)들에 대하여 다차원 인덱스를 구축하고, 인덱싱 공간(feature space)에서 사용하기 위한 새로운 거리 함수 D_{tw_lb}를 사용한다. 이 거리 함수 D_{tw_lb}는 삼각형 부등식 성질을 만족하는 타임 워핑 거리의 하한 함수(lower bound function)이다. 따라서 타임 워핑 거리 대신 D_{tw_lb}를 필터링 단계에서 사용하더라도 착오 기각이 발생하지 않으며, D_{tw_lb}를 기반으로 하는 인덱스를 사용하더라도 인덱스 검색에서 역시 착오 기각이 발생하지 않는다. 이 결과, 이 기법은 착오 기각 없이 고속의 유사 검색을 지원할 수 있다. 본 논문에서는 이 기법을 LB_Filter라 부른다.

본 논문에서는 LB_Filter를 확장함으로써 타임 워핑을 지원하는 *서브시퀀스 매칭*을 착오 기각 없이 효과적으로 처리하는 기법을 제안하고자 한다.

LB_Filter를 응용한 가장 손쉬운 방법은 데이터베이스 내에 존재한 *모든 가능한* 서브시퀀스들에 대하여 특징을 추출하고, 이 특징들에 대하여 다차원 인덱스를 구성하는 것이다. 또한, 서브시퀀스 매칭 시에는 D_{tw_lb}를 이용한 인덱스 검색을 통하여 후보들을 파악하고, 이들에 대한 착오 채택(false alarm) 여부를 조사한다. 이 방법은 LB_Filter를 손쉽게 적용할 수 있다는 장점이 있다. 그러나 길이가 L인 시퀀스 내에서 추출 가능한 모든 서브시퀀스의 수는 (L+1)*L/2로서 매우 많다. 이 결과, 다차원 인덱스 내에 저장해야 하는 특징들의 수는 시퀀스 길이의 제곱에 비례하고, 시퀀스 수에 비례하는 형태를 가진다. 따라서, 대형 데이터베이스 환경을 고려하면, 이 방식은 저장 공간 오버헤드 측면에서 큰 무리가 있다.

이러한 문제를 해결하기 위하여 본 논문에서는 *슬라이딩 윈도우를 기반으로 하는 접두어-질의 기법(prefix-querying method based on sliding windows)*을 제안한다. 제안된 기법에서는 슬라이딩 윈도우(sliding window)[10] 개념을 이용한 인덱싱 전략을 사용한다. 즉, 데이터베이스 내의 각 시퀀스를 고정된 길이의 슬라이딩 윈도우들로 분할하고, 각 윈도우로부터 참고 문헌 [16]의 방식으로 네 개의 특징들을 추출한다. 또한, 이 네 개의 특징들을 인덱싱 애트리뷰트로 이용하여 전체 데이터베이스를 위한 사차원 인덱스를 구성한다.

서브시퀀스 매칭 시에는 질의 시퀀스의 가능한 접두어에 대하여 특징들을 추출하고, 이 특징들과 허용치 ε을 가지고 다차원 인덱스를 검색한다. 인덱스 검색 결과로 반환된 후보들에 대하여 착오

채택 여부를 검사함으로써 최종 질의 결과를 구한다. 제안된 기법의 건고성(robustness)를 규명하기 위하여 서브시퀀스 매칭에서 착오 기각이 발생되지 않음을 증명한다. 또한, 다양한 실험에 의한 성능 분석을 통하여 제안된 기법의 우수성을 제시한다.

본 논문의 구성은 다음과 같다. 제 2장에서는 논의 전개에 필요한 용어 및 기호를 정의한다. 제 3장에서는 타임 워핑 지원 유사 검색에 관한 기존의 연구에 관하여 소개하고, 기존 기법들이 갖는 문제점들을 지적한다. 제 4장에서는 본 연구에서 제안하는 타임 워핑 지원 서브시퀀스 매칭 기법에 관하여 자세히 논의한다. 제 5장에서는 제안하는 기법의 우수성을 규명하기 위한 성능 평가 결과를 제시한다. 끝으로, 제 6장에서는 본 논문을 요약하고, 결론을 내린다.

2. 용어 정의

서픽스 시퀀스 S(= <s[1], s[2], ..., s[|S|]>)는 실수인 요소 값들의 연속이다. 여기서 |S|는 시퀀스의 길이이며, s[i]는 S의 i번째 요소를 의미한다. First(S)와 Last(S)는 각각 S의 첫 번째 요소 s[1]과 마지막 요소 s[|S|]를 의미한다. Rest(S)는 s[1]을 제외한 S의 나머지 요소들로 구성되는 시퀀스 <s[2], s[3], ..., s[|S|]>를 의미한다. S[i:j]는 S의 서브시퀀스로서 i번째 요소에서부터 j번째 요소까지를 포함한다. <>은 요소가 존재하지 않는 널 시퀀스(null sequence)를 의미한다. 데이터베이스 내에 저장된 시퀀스를 데이터 시퀀스라 하고, 유사 검색 질의에 주어지는 시퀀스를 질의 시퀀스라 한다.

길이 n을 갖는 두 시퀀스 S와 Q의 유사한 정도를 측정하기 위하여 다음과 같은 거리 함수 L_p가 널리 사용된다. L_1은 맨하탄 거리(Manhattan distance), L_2는 유클리드 거리(Euclidean distance), L_∞은 대응되는 각 쌍의 거리 중 최대 거리를 의미한다[25]. 거리 함수 L_p는 대상이 되는 두 시퀀스의 길이가 같아야 한다는 제한이 있다.

$$L_p(S,Q) = \left(\sum_{i=1}^{n} | s[i] - q[i] |^p \right)^{1/p}, \quad 1 \le p \le \infty.$$

두 시퀀스 S와 Q간의 타임 워핑 변환을 기반으로 한 타임 워핑 거리(time warping distance) D_{tw}는 다음과 같이 재귀적으로 정의된다[17]:

정의 1:

(1) $D_{tw}(<>, <>) = 0$,

(2) $D_{tw}(S, <>) = D_{tw}(<>, Q) = \infty$,

(3) $D_{tw}(S, Q) = D_{base}(First(S), First(Q)) + \min(D_{tw}(S, Rest(Q)), D_{tw}(Rest(S), Q), D_{tw}(Rest(S), Rest(Q)))$

여기서, min은 인자들 중 가장 작은 값을 가지는 것을 취하는 함수이며, D_{base}는 기본 거리 함수로서 L_p 중 응용에 적합한 임의의 것을 선택하여 사용할 수 있다. 타임 워핑 변환에서는 정의 1(3)에서와 같이 두 시퀀스의 거리 차를 최소화하기 위하여 한 시퀀스 내의 임의의 요소를 반복시킴으로써 이 요소가 다른 시퀀스의 다수의 요소들과 매치되는 것을 허용한다. 이러한 특성으로 인하여 D_{tw}는 요소 추출의 주기가 다르거나 길이가 다른 두 시퀀스의 유사 정도를 측정하는 응용에서 널리 사용된다.

3. 관련 연구

이 본 장에서는 관련 연구로서 타임 워핑 지원 유사 검색에 관한 기존의 접근 방법들[2]을 요약하고, 서브시퀀스 매칭에 적용하는 경우에 발생하는 문제점을 지적한다.

3.1. Naive_Scan

타임 워핑 지원 유사 검색 문제는 음성 인식 분야에서 널리 연구되어 왔으며[17], D_{tw}를 계산하기 위하여 주로 동적 프로그래밍(dynamic programming)을 사용한다[17][4]. 이 기법은 서브시퀀스 매칭을 위하여 데이터베이스 내에 존재하는 모든 서브시퀀스를 액세스하여 각 서브시퀀스 s와 질의 시퀀스 q의 D_{tw}를 계산한다.

이 기법은 다음과 같은 두 가지 원인으로 인하여 높은 CPU 비용을 유발한다. (1) 데이터베이스 내에 평균 길이 L인 M개의 시퀀스들이 존재하는 경우, 가능한 서브시퀀스의 개수는 $O(M*L^2)$로서 매우 많다. (2) 동적 프로그래밍을 이용한 s와 q의 D_{tw} 계산 복잡도(complexity)는 $O(|s|*|q|)$로 매우 높다. 뿐만 아니라, 모든 데이터 시퀀스들을 디스크로부터 액세스해야 하므로 대형 데이터베이스 환경에서는 응답 시간은 매우 길어진다. 본 논문에서는 이 기법을 Naive_Scan이라 부른다.

3.2. LB_Scan

Yi 등[27]은 타임 워핑을 지원하는 전체 매칭 기법을 제안하였다. 이 기법은 두 시퀀스들간의 위치 관계를 이용하여 고안한 하한 함수(lower bound function) D_{lb}를 필터링 단계에서 사용한다.

2) 참고 문헌 [27]에서 다차원 인덱스를 이용한 FastMap 기법을 제안하고 있으나, 이 기법은 착오 기각을 유발하므로 본 논문에서는 이를 관련 연구에서 제외한다.

따라서 착오 기각 없이 최종 질의 결과에 포함될 가능성이 없는 시퀀스들을 효과적으로 제외할 수 있다. D_{lb}의 계산 복잡도는 O(|s|+|q|)이므로 복잡도가 O(|s|*|q|)인 D_{tw}에 비교하여 CPU 비용을 크게 개선할 수 있다.

이 기법은 D_{lb}를 질의 시퀀스와 데이터베이스 내의 모든 가능한 서브시퀀스에 적용함으로써 서브시퀀스 매칭에 쉽게 적용할 수 있다. 그러나 이 기법은 하나의 질의 처리를 위하여 전체 데이터베이스를 디스크로부터 액세스해야 한다는 성능 개선의 한계를 가지므로 대형 데이터베이스 환경에는 적합하지 않다. 본 논문에서는 이 방식을 LB_Scan이라 부른다.

3.3. ST_Filter

참고 문헌 [21]에서는 서픽스 트리(suffix tree)를 인덱스 구조로 이용하는 기법을 제안한 바 있다. 본 논문에서는 이 기법을 ST_Filter라 부른다. ST_Filter에서는 시퀀스들의 서픽스들을 서픽스 트리 내에 저장시킨다. 유사 검색 시에는 이 서픽스 트리를 이용하여 최종 결과에 포함될 후보들을 효과적으로 필터링 한다.

ST_Filter는 전체 매칭 보다는 서브시퀀스 매칭을 주요 응용 대상으로 한다. 데이터베이스 내에 저장된 공통 서브시퀀스들이 많아질수록 서픽스 트리의 성능 개선 효과는 커진다. ST_Filter에서는 많은 공통 서브시퀀스들 생성하기 위하여 도메인 분류(categorization)[21]를 이용한다. 도메인 분류는 요소 값을 심볼로 변환하기 위하여 요소 값의 도메인을 여러 범위들로 분할하는 과정이다. 도메인을 작은 범위들로 분할하면, 트리의 크기가 커지므로 트리 탐색 비용이 증가한다. 반면, 도메인을 적은 수의 큰 범위들로 분할하면, 착오 채택이 많이 발생하므로 후보 시퀀스 액세스 비용이 증가된다. 따라서 실제 응용에 적합한 최적의 도메인 분류를 결정하는 것이 매우 어렵다.

3.4. LB_Filter

참고 문헌 [16]에서는 전체 매칭을 위한 인덱스 기반 유사 검색 기법을 제안하였다. 이 연구에서는 삼각형 부등식을 만족하는 D_{tw}의 새로운 하한 거리 함수 $D_{tw\text{-}lb}$를 고안하였다. $D_{tw\text{-}lb}$는 각 시퀀스 S로부터 타임 워핑과 무관한 네 개의 특징 <First(S), Last(S), Greatest(S), Smallest(S)>를 함수의 인자로서 사용한다. 여기서 First(S), Last(S), Greatest(S), Smallest(S)는 각각 S의 첫 값, 마지막 값, 최대 값, 최소 값을 의미한다.

이러한 특징들을 인자로 사용하는 하한 함수 D_{tw_lb}는 다음과 같이 정의된다.

정의 2:

$D_{tw\_lb}(S, Q) = L_p(Feature(S), Feature(Q))$

여기서, Feature(S) = <First(S), Last(S), Greatest(S), Smallest(S)>, Feature(Q) = <First(Q), Last(Q), Greatest(Q), Smallest(Q)>이며, p는 주어진 응용에서 사용하는 D_{tw}가 채택하는 p와 동일하게 결정된다.

이 기법에서는 이러한 네 특징들을 인덱싱 애트리뷰트로 사용하는 사차원 인덱스를 구성하고, 이 인덱스와 $D_{tw\text{-}lb}$를 이용한 필터링 단계를 통하여 전체 매칭을 매우 효율적으로 처리한다. 본 논문에서는 이 기법을 LB_Filter라 부른다.

LB_Filter를 서브시퀀스 매칭에 응용하기 위한 가장 단순한 방법은 데이터베이스 내에 존재한 *모든 가능한* 서브시퀀스들에 대하여 네 개의 특징들을 추출하고, 이들에 대하여 다차원 인덱스를 구성하는 것이다. 서브시퀀스 매칭 시에는 이 인덱스를 이용하여 후보들을 효과적으로 파악할 수 있다. 그러나 길이가 L인 시퀀스 내에 존재하는 가능한 서브시퀀스의 수는 (L+1)*L/2로서 매우 많다. 이 결과, 다차원 인덱스 내에 저장해야 하는 특징들의 수는 시퀀스 길이의 이차 함수로 증가하고, 시퀀스 수의 일차 함수로 증가하는 형태를 가진다. 따라서 대형 데이터베이스 환경에서는 저장 공간 오버헤드 측면에서 큰 무리가 있다[3].

4. 제안하는 기법

완전 먼저, 본 논문에서 해결하고자 하는 타임 워핑 기반 서브시퀀스 매칭 문제를 다음과 같이 정의한다.

문제 정의:

m개의 데이터 시퀀스 $S_1, S_2, ..., S_m$로 구성된 시퀀스 데이터베이스, 질의 시퀀스 q, 유사 허용치 ε가 주어질 때, 데이터베이스로부터 q와의 D_{tw}가 ε이하인 서브시퀀스들을 반환한다.

본 장에서는 이러한 문제를 해결하기 위한 효과적인 기법을 제시한다. 제안하는 기법이 추구하는 목표는 정확한 질의 결과, 빠른 검색 성능, 그리고 작은 저장 공간 비용을 보장하는 것이다.

먼저, 제 4.1절에서는 제안하는 기법에서 채택하는 타임 워핑 기반 유사 모델을 설명하고, 채택 배경에 관하여 설명한다. 제 4.2절에서는 타임 워핑

3) 예를 들어, 데이터베이스 내에 평균 길이 1,000인 1,000개의 시퀀스들이 저장되어 있는 경우 500,500,000개의 특징들을 포함하는 사차원 인덱스를 구성해야 한다.

기반 서브시퀀스 매칭을 위한 효과적인 인덱싱 전략을 제안한다. 제 4.3절에서는 제안하는 인덱싱 전략을 기반으로 한 서브시퀀스 매칭 알고리즘을 제안한다. 제 4.4절에서는 인덱스 검색의 회수를 줄임으로써 추가로 성능을 개선시키는 방안에 관하여 논의한다.

4.1. 유사 모델

본 연구에서는 두 시퀀스들의 유사한 정도를 나타내는 척도로서 정의 1에 나타난 타임 워핑 거리 D_{tw}를 사용한다. 특히, 요소 반복을 통하여 변환된 두 시퀀스간의 거리 함수 D_{base}로서 L_∞를 사용한다. 이를 위하여 타임 워핑 거리는 다음 정의 3과 같은 형태로 변형된다.

정의 3:

(1) $D_{tw}(<>, <>) = 0$,

(2) $D_{tw}(S, <>) = D_{tw}(<>, Q) = \infty$,

(3) $D_{tw}(S, Q) = \max(|First(S) - First(Q)|,$
$\min D_{tw}(S, Rest(Q)), D_{tw}(Rest(S), Q),$
$D_{tw}(Rest(S), Rest(Q)))$

D_{base}로서 L_1을 사용하는 참고 문헌 [4][27][21]과 달리 본 연구에서 L_∞를 사용하는 주된 이유는 사용자의 질의 작성의 부담을 덜도록 하기 위해서이다. 정의 1에서 나타난 바와 같이 L_1을 사용하는 경우, 타임 워핑 거리는 변환된 두 시퀀스의 각 대응되는 요소 쌍의 거리들의 합으로 나타나므로 질의 시퀀스와 데이터 시퀀스의 길이에 큰 영향을 받는다. 본 논문에서 고려하는 서브시퀀스 매칭 문제에서는 같은 데이터 시퀀스로부터 추출되는 서브시퀀스들이라 할 지라도 그 길이 차가 매우 크므로 이러한 문제는 더욱 심각하게 나타난다[4]. 따라서 질의를 작성하는 사용자가 해당 데이터베이스 특성에 맞는 적절한 ε을 결정한다는 것은 매우 어려운 일이다. 특히, 동적인 환경에서는 시퀀스의 길이가 계속 변경되므로 올바른 ε를 결정하는 것이 사실상 불가능하다. 반면, L_∞를 사용하는 경우, 시퀀스의 길이에 영향을 받지 않고 일관된 ε을 사용할 수 있으므로 이러한 질의 작성의 부담을 덜 수 있다.

그러나 이와 같은 L_∞의 사용은 사용자의 질의 작성 부담을 덜기 위한 유사 모델상의 변화를 의미하며, 이후에 제안되는 인덱스 기반 서브시퀀스 매칭 기법이 이 유사 모델에 종속되는 것은 아니다. 따라서 D_{base}로서 L_1혹은 L_2를 사용하는 경우에도 본 논문에서 제안하는 인덱스 기반 서브시퀀스 매칭 기법은 그대로 사용할 수 있음을 밝혀두고자 한다.

4.2. 인덱싱 전략

본 논문에서는 효과적인 서브시퀀스 매칭을 지원하기 위하여 *슬라이딩 윈도우 기반 접두어-질의 기법(prefix-querying method based on sliding windows)*을 제안한다. 이 기법은 착오 기각 배제를 위한 이론적 배경으로서 다음과 같은 정리를 기반으로 한다.

정리 1:

임의의 두 시퀀스 s, q, 그리고 임의의 양수 w $(1 \le w \le |s|)$에 대하여, 만일 s와 q의 타임 워핑 거리가 ε이내이면, s의 접두어 s[1:w]와의 타임 워핑 거리가 ε이내인 q의 접두어가 반드시 존재한다. 즉, 아래의 공식이 성립한다.

$$(\exists \boldsymbol{x})\ D_{tw}(s, q) \le \varepsilon \Rightarrow (D_{tw}(s[1{:}w], q[1{:}\boldsymbol{x}]) \le \varepsilon)$$

증명:

$p = <p[1], p[2], ..., p[|p|]>$를 두 시퀀스 s와 q의 D_{tw}를 최소화하는 워핑 경로(warping path)[4]라 하자. 여기서, $|s| \le |p|$ 이고, $|q| \le |p|$이다. 또한, 워핑 경로의 각 요소를 $p[h] = (s[i_h], q[j_h])$라 하자. 또한, 워핑 경로 내에 존재하는 s와 q의 타임 워핑된 시퀀스를 각각 $s'(=<s[i_h]>)$, $q'(=<q[j_h]>)$라 하자. 여기서, $1 \le h \le |p|$, $1 \le i_h \le |s|$, $1 \le j_h \le |q|$ 이다. 이때, 두 시퀀스 s와 q의 D_{tw}는 다음과 같이 계산된다.

$$D_{tw}(s, q) = L_\infty(s', q')$$

여기서, 워핑 경로의 단조 및 연속 성질(monotonic and continual property)[4]에 따라 다음의 식이 만족된다.

$(\exists x)$ $(p[x] = (s[w], q[jx]),$ 여기서 $1 \le w \le |s|$, $1 \le x \le |p|$, $1 \le jx \le |q|$.

L_∞를 사용하는 D_{tw}는 워핑 경로 내에 존재하는 두 요소 값의 차의 최대 값을 취하므로, p[1]부터 p[x]까지의 서브 워핑 경로는 워핑 경로 p보다 항상 작은 타임 워핑 거리를 반환한다. 따라서 워핑 경로 p가 ε이하의 거리를 반환하는 경우, p[1]부터 p[x]까지의 서브 워핑 경로도 역시 ε이하의 거리를 반환한다. p[1]부터 p[x]까지의 서브 워핑 경로의 거리가 $D_{tw}(s[1{:}w], q[1{:}j_x])$를 의미하므로 위의 정리는 성립한다.

---

4) 예를 들어, 길이가 1000인 데이터 시퀀스로부터 추출되는 서브시퀀스는 최소 1에서부터 최대 1000까지의 다양한 길이를 가질 수 있다. 길이가 10인 질의 시퀀스 q, 길이가 1인 서브시퀀스 s1, 길이 1000인 서브시퀀스 s2를 고려해 보자. 만일, L_1 혹은 L_2를 D_{base}로 사용하는 경우, 유사성을 판별하기 위하여 $D_{tw}(q, s1)$과 $D_{tw}(q, s1)$를 같은 ε값을 기준으로 비교한다는 것은 매우 불합리하다.

```
TW_Subseq_Matching1
Input: query sequence q, tolerance ε
Output: a set of pairs (sequence identifier, subsequence position)
candSet = {};
ansSet = {};

FOR each j between minPrefixLen and maxPrefixLen DO
(1) Extract four features from q[1:j];
(2) queryResult = PerformRangeQuery(<features for q[1:j]>, ε);
(3) candSet = candSet ∪ queryResult;
FOR each entry in candSet DO
(4) Read the corresponding subsequence s from the database;
(5) Build a cumulative distance table T between q and s;
(6) IF Dtw(q, s) < ε, insert q into ansSet;
    return ansSet;
```

알고리즘 1. 타임 워핑 기반 서브시퀀스 매칭 알고리즘 TW_Subseq_Maching1.

LB_Filter에서 사용하였던 거리 함수 D_{tw_lb}는 타임 워핑 거리 D_{tw}의 하한 함수이므로, 정리 1로부터 다음과 같은 따름 정리 1을 쉽게 유도할 수 있다.

따름 정리 1:

임의의 두 시퀀스 s, q, 그리고 임의의 양수 w ($1 \le w \le |s|$)에 대하여, 다음의 식이 항상 성립한다. : $(\exists x)\ Dtw(s, q) \le \varepsilon \Rightarrow (Dtw\_lb(s[1:w], q[1:x]) \le \varepsilon)$, 여기서 $1 \le j \le |q|$.

따름 정리 1을 효과적인 서브시퀀스 매칭에 활용하기 위하여 본 논문에서는 슬라이딩 윈도우(sliding window)[10] 개념을 이용한다. 즉, 데이터베이스 내의 각 시퀀스를 길이가 w인 슬라이딩 윈도우들로 분할하고, 각 윈도우로부터 네 개의 특징 First(w), Last(w), Greatest(w), Smallest(w)를 추출한다. 이 네 개의 특징들을 인덱싱 애트리뷰트로 이용하여 전체 데이터베이스를 위한 사차원 인덱스를 구성한다. 길이가 L인 시퀀스로부터 추출되는 길이가 w인 슬라이딩 윈도우의 수는 (L-w+1) 개이다. 이것은 모든 가능한 서브시퀀스의 수 (L+1)*L/2와 비교하여 훨씬 작은 수이므로 인덱스의 저장 공간 측면에서 실용 가능하다.

윈도우의 길이 w는 서브시퀀스 매칭의 처리 성능과 매우 밀접한 관계가 있다. 크기 w인 슬라이딩 윈도우들에 대하여 인덱스가 구성되어 있다고 가정하자. 서브시퀀스 s, 질의 시퀀스 q에 대하여, w의 크기가 작을수록 $D_{tw}(s, q)$와 $D_{tw\_lb}(s[1:w], q[1:x])$ ($x \le |q|$)의 차이는 커지며, 이 결과, D_{tw_lb}를 이용한 인덱스 필터링 단계에서 반환되는 착오 채택의 수는 많아진다. 또한, q와 w의 차이가 커질수록 s[1:w]와 비교해야 하는 q의 접두어의 개수가 많아진다. 이 두 가지 사실은 윈도우의 길이가 질의 시퀀스 길이 보다 작을수록 서브시퀀스 매칭의 처리 성능이 저하됨을 의미하는 것이다. 반대로, 윈도우의 길이가 질의 시퀀스의 길이보다 큰 경우에는 인덱스를 이용한 필터링이 불가능하다.

이러한 두 가지 문제를 해결하기 위한 합리적인 대안으로서 minQLen과 maxWarpRatio[Ber96]의 두 가지 개념을 사용한다. minQLen은 응용에서 사용될 수 있는 최소의 질의 시퀀스 길이를 의미하며, maxWarpRatio는 타임 워핑 변환에 의하여 하나의 시퀀스 요소가 반복될 수 있는 최대 수를 의미한다[4]. 예를 들어, maxWarpRatio가 3일 때, 시퀀스 s의 타임 워핑 변환된 s'의 길이는 |s|와 3|s|사이가 된다. 본 연구에서는 응용의 특성에 따라 minQLen과 maxWarpRatio가 미리 결정된다는 가정 하에, 윈도우의 크기를 $w = \lceil \frac{\text{min_QLen}}{\text{max_WarpRatio}} \rceil$로 결정한다. w는 최소 길이의 질의 시퀀스와 매치될 수 있는 데이터베이스 내의 최소 서브시퀀스의 길이를 의미하는 것이다.

이러한 크기 w의 윈도우들을 대상으로 구성된 인덱스를 이용하여 질의 시퀀스 q를 위한 서브시퀀스 매칭은 다음과 같이 처리된다. 먼저, 인덱스 내에 저장된 크기 w인 서브시퀀스(즉, 윈도우)와 매치될 수 있는 q의 접두어의 최소 및 최대 길이를 결정한다. 이러한 접두어의 최소 길이

minPrefixLen은 $\lceil \frac{w}{\max\ WarpRatio} \rceil$ 이며, 최대 길이 maxPrefixLen은 w*maxWarpRatio = minQLen 이다. 다음으로, q의 각 접두어 q[1:j] (minPrefixLen≤j≤maxPrefixLen)에 대하여, 네 개의 특징 First(q[1:j]), Last(q[1:j]), Greatest(q[1:j]), Smallest(q[1:j])를 추출하고, 이 특징들과 D_{tw_lb}, 그리고 ε을 이용하여 인덱스 검색을 수행한다. 끝으로, 인덱스 검색을 통하여 반환된 후보들을 실제로 액세스하여 착오 채택을 제거한다.

4.3.알고리즘

본 절에서는 제 4.2절에서 논의한 전략을 기반으로 인덱스를 생성하는 알고리즘과 질의를 처리하는 알고리즘을 제시한다.

4.3.1. 인덱스 구성

데이터베이스로부터 얻어진 길이 w의 모든 가능한 서브시퀀스(즉, 슬라이딩 윈도우)로부터 네 개의 특징을 추출하므로, 각 윈도우는 사차원 유클리드 공간상의 점으로 표현된다. 따라서 이러한 특징들을 효과적으로 검색하기 위한 인덱스 구조로서 R-트리[13], R+-트리[24], R*-트리[3], X-트리[5] 등 다차원 인덱스를 사용한다. 인덱스 구성 알고리즘은 우선 각 슬라이딩 윈도우 w를 액세스하여 인덱스 엔트리 <First(w), Last(w), Greatest(w), Smallest(w), Loc(w)>를 구한 후, 이 엔트리를 사차원 인덱스 내에 삽입함으로써 인덱스를 구성한다. 여기서 L_{oc}(w)는 w가 속하는 시퀀스 S의 식별자(identifier)와 S내에서의 w의 시작 위치를 나타낸다. 많은 데이터 시퀀스들이 이미 존재하는 경우에는 이러한 반복 삽입 방식이 아닌 벌크 로딩 기법(bulk loading)[14]을 이용함으로써 보다 효과적인 방법으로 다차원 인덱스를 구성할 수 있다.

4.3.2. 질의 처리

알고리즘 1은 사차원 인덱스를 이용하여 질의 시퀀스 q와의 D_{tw}가 ε 이내인 유사한 서브시퀀스들을 데이터베이스로부터 검색하는 알고리즘 TW_Subseq_Maching1을 나타낸 것이다. 이 알고리즘은 길이가 minPrefixLen에서 maxPrefixLen 사이인 q의 접두어에 대하여 인덱스 검색을 수행한다. 일치 규칙 매칭 문제는 다음과 같이 정의된다.

단계 (1)에서는 질의 시퀀스 q의 접두어 q[1:j]로부터 네 개의 특징들을 추출하고, 단계 (2)에서는 사차원 인덱스를 이용한 정사각형 형태의 영역 질의(range query)를 수행한다. 이때, 단계 (1)에서 구한 네 특징들은 영역 질의의 중심점이 되며, ε은 질의의 범위가 된다. 또한, 거리 함수로는 D_{tw_lb}가 사용된다. 단계 (3)에서는 영역 검색의 결과로 반환된 엔트리들로 후보 집합을 구성한다. 단계 (4)~(6)은 후보 집합에 포함된 각 엔트리와 대응되는 시퀀스들에 대하여 착오 채택 여부를 판정하는 것이다. 단계 (4)에서는 후보 집합에 포함된 각 엔트리와 대응되는 윈도우를 접두어로 포함하는 가장 긴 서브시퀀스 s를 데이터베이스로부터 읽어들인다. 단계 (5)에서는 s와 q간의 타임 워핑 거리를 계산하기 위한 누적 거리 테이블(cumulative distance table)[4]을 구성한다. 이때, 계산 효율을 높이기 위하여 참고 문헌 [21]에서 사용하는 가지치기 방식(pruning method)를 사용한다. 단계 (6)에서는 누적 거리 테이블을 이용하여 계산한 이 시퀀스와 질의 시퀀스 Q간의 실제 D_{tw}가 ε이하이면 이를 최종 결과로 반환한다.

4.4. 인덱스 검색 성능의 개선

제안된 질의 처리 알고리즘에서는 질의 시퀀스 q의 각 접두어 q[1:j] (minPrefixLen≤j≤maxPrefixLen)에 대하여 개별적으로 인덱스 검색을 수행한다. 따라서 하나의 서브시퀀스 매칭의 처리를 위한 인덱스 검색의 수는 maxPrefixLen-minPrefixLen+1회이다. 이 값이 커지는 경우, 인덱스 검색의 비용이 커지므로 서브시퀀스 매칭의 처리 성능이 저하될 수 있다. 제 4.2절에 나타난 정의에 의하면, minPrefixLen과 maxPrefixLen의 차는 maxWarpRatio가 커짐에 따라 커진다. 따라서 maxWarpRatio의 값이 큰 경우에도 좋은 서브시퀀스 매칭의 성능을 얻기 위해서는 인덱스 검색의 회수를 줄이는 전략이 필요하다.

본 논문에서는 각각의 접두어에 대하여 개별적으로 인덱스 검색을 수행하지 않고, 그룹으로 병합한 다수의 접두어들에 대하여 한번의 인덱스 검색을 수행하는 방법을 사용한다. <F, L, G, S>와 <F′, L′, G′, S′>를 각각 질의 시퀀스 q의 접두어 q[1:j]와 q[1:j′] (minPrefixLen≤j≤ maxPrefixLen, minPrefixLen≤j′≤maxPrefixLen)의 네 특징들이라 하자. 유사 허용치가 ε이라 할 때, 두 개의 접두어와 대응되는 인덱스 검색에서 사용되는 질의 영역은 각각 ([F-ε, F+ε], [L-ε, L+ε], [G-ε, G+ε], [S-ε, S+ε])와 ([F′-ε, F′+ε], [L′-ε, L′+ε], [G′-ε, G′+ε], [S′-ε, S′+ε])이다. 이 두 개의 질의 영역들을 하나의 질의 영역으로 병합하기 위하여 먼저 <F, L, G, S>와 <F′, L′, G′, S′>를 모두 포함하는 최소의 사각형(minimum bounding rectangle: MBR)을 형성한 후, 형성된

MBR의 각 차원의 최소 값과 최대 값을 ε만큼 확장한다. <F, L, G, S>와 <F′, L′, G′, S′>을 위한 MBR은 ([min(F,F′), max(F,F′)], [min(L,L′), max(L,L′)], [min(G,G′), max(G,G′)], [min(S,S′), max(S,S′)])이 되며, 최종 확장된 질의 영역은 ([min(F,F′)-ε, max(F,F′)+ε], [min(L,L′)-ε, max(L,L′)+ε], [min(G,G′)-ε, max(G,G′)+ε], [min(S,S′)-ε, max(S,S′)+ε])이 된다.

이러한 과정을 일반화함으로써 두 개 이상의 질의 영역들을 하나로 병합시킬 수 있다. 극단의 경우, 질의 시퀀스의 모든 접두어들과 대응되는 질의 영역들을 하나의 질의 영역으로 만들 수 있다. 이 경우, 인덱스 검색을 위한 비용을 크게 줄일 수 있다. 그러나 이러한 병합의 결과 생성되는 MBR은 일반적으로 원래의 MBR들의 합보다 커지게 되므로 후보 집합을 크게 하는 경향이 있다. 이 결과, 최종 질의 결과를 위한 착오 채택 판정 비용이 커지게 된다. 따라서 인덱스 검색 비용과 착오 채택 판정 비용을 동시에 최적화 할 수 있는 MBR 구성 전략이 요구된다.

최종 질의 영역을 위한 MRB 구성 방법을 요약하면 다음과 같다.

(1) 각 접두어와 일대일 대응되는 다수의 질의 MBR들을 구성하는 방법
(2) 모든 접두어들을 포함하는 단 하나의 질의 MBR을 구성하는 방법
(3) 접두어들을 다수의 그룹들로 분할하고, 각 그룹과 대응되는 MBR을 다수 구성하는 방법[5)]

일반적으로 접두어의 수가 많아지는 경우에는 MBR의 지나친 크기 증가를 억제하기 위하여 다수의 MBR로 구성하는 것이 효과적이다. 그러나 같은 질의 시퀀스로부터 추출된 접두어들은 공통점이 많으므로, 대응되는 특징 벡터들이 사차원 특징 공간에서 인접한 위치에 존재할 가능성이 높다. 따라서 이러한 특징 벡터들을 하나의 MBR로 표현하는 경우에도, 그 영역의 크기가 비정상적으로 증가하지는 않는다. 이러한 MBR 구성에 관한 사항은 본 연구의 주요 초점이 아니므로, 본 논문에서는 전체 문제를 단순화하기 위하여 방법 (1)과 방법 (2)만을 대상으로 하고, 구체적인 MBR 구성 방안에 관해서는 향후 연구로 다루고자 한다. 본 논문에서는 방법 (2)를 채택한 개선된 질의 처리 알고리즘 TW_Subseq_Maching2라 부른다.

5) 본 연구에서는 방법 (3)을 위하여 참고 문헌 [10]의 I-adaptive 방식에서 사용하는 방법을 향후 연구로 고려하고 있다. I-adaptive 방식은 데이타 시퀀스로부터 추출된 다수의 슬라이딩 윈도우들을 다수의 MBR들로 그룹화 하는 휴리스틱 방법이다. 이 방식은 현재 구성 중인 MBR에 새로운 윈도우가 포함되는 경우의 추가 비용을 추정하기 위한 비용 공식(cost formula)[15]을 사용함으로써 인덱스 검색 비용과 객체 액세스 비용을 동시에 최적화 하는 것을 목표로 한다.

5. 성능 분석

본 장에서는 실험에 의한 성능 분석을 통하여 제안하는 기법의 우수성을 규명한다. 제 5.1절에서는 실험 환경을 설명하고, 제 5.2절에서는 실험 결과를 분석한다.

5.1. 실험 환경

본 연구에서는 합성 데이터 Syn_Data와 실제 데이터 S&P_Data를 이용한 실험을 수행한다. Syn_Data내의 각 시퀀스 S = <s[1], s[2], ..., s[n]> 는 다음과 같은 랜덤 워크(random walk) 형태를 가진다.

$$s[i] = s[i-1] + z[i]$$

여기서 z[i]는 구간 [-0.1, 0.1] 사이에서 균일한 분포를 취하는 랜덤 변수이며, 시퀀스의 첫 요소 값 s[1]은 구간 [1, 10] 사이의 임의의 값을 취하는 랜덤 변수이다. 실제 데이터 S&P_Data는 미국의 S&P 500 주식 데이터이며[6)], 평균 길이가 231인 545개의 시퀀스들로 구성된다.

각 실험을 위하여 다음과 같이 생성된 질의 시퀀스를 가지는 100개의 서브시퀀스 매칭을 수행한다: (1) 데이터베이스로부터 임의로 하나의 시퀀스를 선택한다; (2) 선택된 시퀀스로부터 임의의 서브시퀀스를 무작위로 추출한다; (3) 추출된 서브시퀀스의 각 요소 값을 위한 적절한 범위[7)] 내의 임의의 값을 선택한다; (4) 선택된 값을 대응되는 요소 값에 더한다. 본 실험에서는 성능 평가 지수로서 100개의 서브시퀀스 매칭의 결과로 나타난 평균 수행 시간(average elapsed time)을 사용한다.

본 실험을 위한 하드웨어 플랫폼으로는 SunOS 5.8을 운영체제로 사용하고 640MB의 주기억장치를 갖는 SunSparc Ultra-5 워크스테이션을 사용한다. 장착된 하드디스크는 9.5ms의 탐색 시간을 가지는 Seagate ST39140A이며, 그 크기는 9GB이다. 실험 중 다른 사용자 및 데몬 프로세스의 상호 간섭을 방지하기 위하여 운영 체제를 단일 사용자용으로 설정한 후 실험한다.

6) 이 S&P 500 주식 데이타는 웹 싸이트 http://biz.swcp.com/stocks에서 구할 수 있다.
7) 선택된 시퀀스 내에 속하는 요소 값들의 표준편차를 std라 할 때, 이 범위는 [-std/10, std/10]이다.

표 5.1. 실험에서 사용된 인자 값.

	exp 1	exp 2	exp 3	exp 4	exp 5	exp 6
data set	S&P 500	S&P 500	S&P 500	S&P 500	synthetic	synthetic
seqNum	545	545	545	545	1000-4000	500
sepLen	231	231	231	231	200	200-800
avgQLen	90	90	90	50-110	90	70-280
minQLen	50	50	30-90	50	50	50-200
maxWarpRatio	5	3-9	5	5	5	5
window size	10	6-17	6-18	10	10	10-40
ε	0.2-1.2	0.6	0.6	0.6	0.1	0.1

성능 평가를 위하여 제안된 TW_Subseq_Matching1와 TW_Subseq_Matching2의 성능을 기존의 LB_Scan[27], ST_Filter[21]의 성능과 비교 검토한다. 동일한 상황에서의 성능 평가를 위하여 본 실험에서는 D_{tw}를 위한 기본 거리 함수 D_{base}로서 L_1를 사용하는 기존의 기법도 본 논문에서 채택하는 L_∞을 사용하도록 한다. 또한, ST_Filter를 위한 도메인 분류 방법으로서 최대 엔트로피 기법(maximum entropy method)을 이용하여 ST_Filter가 50개의 구간을 갖도록 한다. TW_Subseq_Matching1와 TW_Subseq_Matching2를 위한 다차원 인덱스로는 현재 가장 널리 채택되고 있는 R^*-트리[3]를 사용하여 실험한다. 사용된 페이지 크기는 1KB이다. 표 5.1은 각 실험에서 사용된 인자들의 값을 요약한 것이다.

5.2. 실험 결과 및 분석

먼저, 실험 1에서는 허용치 ε를 0.2, 0.4, 0.6, 0.8로 변화하면서 각 기법의 성능을 비교하였다[8]. 사용된 다른 인자로서 maxWarpRatio, minQLen, avgQLen을 5, 50, 90으로 각각 설정하였다. 여기서 aveQLen은 평균 질의 시퀀스의 길이를 나타낸다. 그림 5.1은 S&P_Data에 대한 각 기법의 실험 결과를 나타낸 것이다. 여기서 세로 축은 밑이 10인 로그 스케일을 사용한다.

실험 결과에 의하면, 허용치가 커짐에 따라 모든 기법들의 처리 시간은 증가하는 것으로 나타났다. 먼저, LB_Scan과 ST_Filter를 비교해 보자. 허용치가 0.2, 0.4, 0.6으로 작은 경우에는 ST_Filter가 좋은 성능을 보였으나, 허용치가 0.8, 1.0, 1.2로 큰 경우에는 LB_Scan이 더 좋은 성능을 보였다. 이러한 현상의 주요 원인은 접미어 트리의 대형화에 따른 것이다. 일반적으로, 접미어 트리의 크기는 대상인 데이터베이스 자체의 크기보다 훨씬 커지게 된다[16]. 이 경우에도 허용치가 작으면 ST_Filter는 접미어 트리의 극히 일부만을 액세스하게 되므로 좋은 검색 성능을 보이게 된다. 그러나 허용치가 커지는 경우에는 ST_Filter가 대용량의 접미어 트리의 많은 부분을 액세스해야 하므로 성능이 크게 저하된다.

그림 5.1에 나타난 바와 같이, 제안된 두 기법은 기존의 LB_Scan과 ST_Filter에 비하여 뛰어난 검색 성능을 보였다. TW_Subseq_Matching1와 TW_Subseq_Matching2의 검색 성능을 비교해 보자. 허용치가 작은 0.2, 0.4, 0.6의 경우, TW_Subseq_Matching1이 TW_Subseq_Matching2보다 낮은 성능을 보인다. 그러나 허용치가 큰 0.8, 1.0, 1.2의 경우에는 그 반대의 결과가 나타났다. 이는 큰 허용치로 인한 단일 질의 MBR의 추가적인 착오 채택의 수는 상대적으로 작은 반면, 단 한번의 인덱스 검색으로 인한 비용 절감의 효과는 매우 크기 때문이다. 허용치가 0.2인 경우, TW_Subseq_Matching1은 기존 기법 중 나은 성능을 가지는 ST_Filter와 비교하여 약 42배 빠른 검색 성능을 보였으며, 허용치가 1.2인 경우, TW_Subseq_Matching2는 기존 기법 중 나은 성능을 가지는 LB_Scan과 비교하여 약 6배 빠른 검색 성능을 보였다.

실험 2에서는 maxWarpRatio를 3, 5, 7, 9로 변화하면서 각 기법의 성능을 비교하였다. 사용된 다른 인자로서 ε, minQLen, avgQLen을 0.6, 50, 90으로 각각 설정하였다. 그림 5.2는S&P_Data에 대한 각 기법의 실험 결과를 나타낸 것이다. maxWarpRatio가 증가함에 따라 모든 기법들의 검색 시간이 점차적으로 증가하는 것으로 나타났다. 이것은 maxWarpRatio가 증가함에 따라 최종 결과

8) 허용치가 0.2인 경우는 평균 8개의 최종 결과를 반환하였고, 허용치가 1.2인 경우는 평균 1703개의 최종 결과를 반환하였다.

로 리턴되는 서브시퀀스들의 수뿐만 아니라 후보 시퀀스들의 수도 많아지기 때문이다. TW_Subseq_Matching1은 모든 maxWarpRatio 값들에 대하여 TW_Subseq_Matching2에 비하여 약간 좋은 성능을 보였으며, 기존 기법 중 나은 것과 비교해서는 maxWarpRatio가 3인 경우에는 35 배, maxWarpRatio가 9인 경우에는 21배의 성능 개선 효과를 보였다.

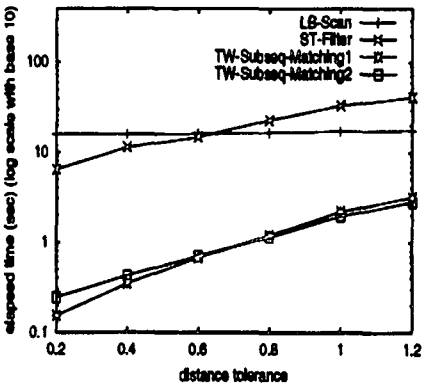


그림 5.1. 허용치 ε의 변화에 따르는 평균 수행 시간의 비교(S&P_Data).

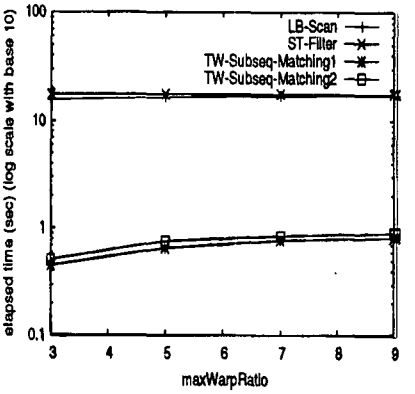


그림 5.2. maxWarpRatio의 변화에 따르는 평균 수행 시간의 비교(S&P_Data).

실험 3에서는 minQLen을 30, 50, 70, 90으로 변화하면서 각 기법의 성능을 비교하였다. 사용된 다른 인자로서 ε, maxWarpRatio, avgQLen을 0.6, 5, 90으로 각각 설정하였다. 그림 5.3은 S&P_Data에 대한 각 기법의 실험 결과를 나타낸 것이다. 기존의 두 기법들은 minQLen의 값에 영향을 받지 않고 일정한 검색 성능을 보인 반면, 제안된 기법들은 minQLen이 증가함에 따라 검색 성능이 좋아지는 것으로 나타났다. 이것은 minQLen이 커질수록 질의 시퀀스의 접두어 수가 줄어들기 때문이다. 이 결과, TW_Subseq_Matching1의 경우에는 인덱스 검색의 수가 줄어들며, TW_Subseq_Matching2의 경우에는 착오 채택의 수가 줄어든다. 기존의 기법들과 비교하여 TW_Subseq_Matching1은 minQLen이 30인 경우에는 23배, minQLen이 90인 경우에는 28배 빠른 검색 성능을 보였다.

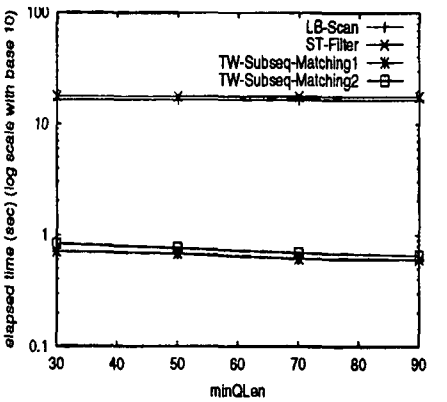


그림 5.3. minQLen의 변화에 따르는 평균 수행 시간의 비교(S&P_Data).

실험 4에서는 avgQLen을 50, 70, 90, 110으로 변화하면서 각 기법의 성능을 비교하였다. 사용된 다른 인자로서 ε, maxWarpRatio, minQLen을 0.1, 5, 50으로 각각 설정하였다. 그림 5.4는 S&P_Data에 대한 각 기법의 실험 결과를 나타낸 것이다. 실험 결과에 의하면, 모든 기법들이 avgQLen 값의 변화에 영향을 받지 않고, 일정한 검색 성능을 나타냈다. 이러한 현상은 다음과 같이 해석될 수 있다: (1) 질의 시퀀스의 길이가 길어질수록 계산 비용은 증가하지만, D_{base}로서 L_∞를 사용하는 경우 이러한 증가는 대단하지 않다. (2) maxWarpRatio가 고정된 경우에는 질의 시퀀스가 길어질수록 최종 결과로 반환되는 서브시퀀스의 수는 줄어든다. TW_Subseq_Matching1의 검색 성능은 기존 기법들과 비교하여 25배 빠른 것으로 나타났다.

S&P_Data는 비교적 소규모의 데이터베이스이다. 이후의 실험에서는 제안된 기법의 대규모 데이터베이스에서도 효과적으로 동작함을 규명하기 위하여 다양한 시퀀스의 수 및 시퀀스 길이를 갖는 대규모의 Syn_Data를 이용하여 대규모 데이터베이스 환경에서 ST_Filter 보다 나은 성능을 가진다고 알려진 LB_Scan[Kim00]과 제안된 기법의 성능을 비교한다.

실험 5에서는 데이터 시퀀스들의 수를 1000, 2000, 3000, 4000으로 변화하면서 각 기법의 성능

을 비교하였다. 사용된 다른 인자로서 ε, maxWarpRatio, minQLen, avgQLen을 0.1, 5, 50, 90으로 각각 설정하였다. 또한, 질의 시퀀스 길이는 200으로 설정하였다. 그림 5.5는 Syn_Data에 대한 각 기법의 실험 결과를 나타낸 것이다. 시퀀스의 수가 증가함에 따라 각 기법의 검색 시간은 선형적으로 증가하는 것으로 나타났다. 그 이유는 (1) 윈도우의 크기, 유사 허용치, maxWarpRatio가 고정된 상황에서는 검색 성능은 최종 결과로 반환되는 서브시퀀스의 수에만 영향을 받으며, (2) 평균적으로 최종 결과 시퀀스들의 수는 데이터베이스 내의 시퀀스 수에 선형적으로 비례하기 때문이다. 실험 결과에 의하면, TW_Subseq_Matching1의 검색 성능은 LB_Scan의 검색 성능과 비교하여 1,000개의 시퀀스들에 대하여 19배, 4,000개의 시퀀스들에 대하여 13배 좋은 것으로 나타났다.

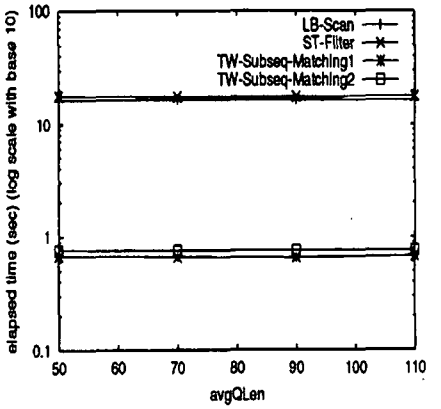


그림 5.4. avgQLen의 변화에 따르는 평균 수행 시간의 비교(S&P_Data).

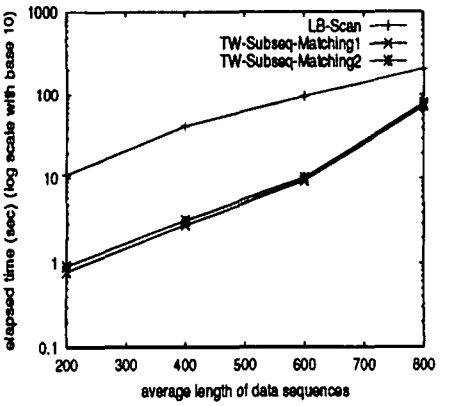
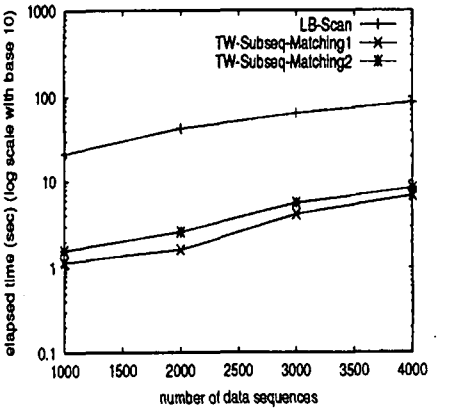


그림 5.5. 데이터 시퀀스 수의 변화에 따르는 평균 수행 시간의 비교(Syn_Data).

그림 5.6. 데이터 시퀀스 길이의 변화에 따르는



평균 수행 시간의 비교(Syn_Data).

실험 6에서는 데이터 시퀀스의 길이를 200, 400, 600, 800으로 변화하면서 각 기법의 성능을 비교하였다. 사용된 다른 인자로서 데이터 시퀀스의 수, ε, maxWarpRatio를 각각 500, 0.1, 5로 각각 설정하였다. minQLen은 각 데이터 시퀀스 길이의 1/4로 설정하였으며, avgQLen은 각각 (minQLen + minLen*(2/5))으로 설정하였다. 또한, 질의 시퀀스 길이로는 500을 설정하였다. 그림 5.6은 에 나타난 바와 같이, 검색 성능의 경향은 그림 5.5에 나타난 것과 매우 유사하며, 제안된 기법은 긴 시퀀스들을 가지는 데이터베이스에서도 매우 효과적인 것으로 나타났다.

6. 결론

유사 검색이란 주어진 질의 시퀀스와 변화의 패턴이 유사한 시퀀스들을 시퀀스 데이터베이스로부터 찾아내는 연산이며[1][10], 데이터 마이닝 및 데이터 웨어하우징 분야에서 널리 사용된다[23]. 타임 워핑은 데이터베이스내의 시퀀스들의 길이가 서로 달라서 유클리드 거리를 이용하여 유사 정도를 직접 측정할 수 없는 경우에 매우 유용한 변환이다[27][21]. 본 논문에서는 타임 워핑을 지원하는 유사 검색을 처리하기 위한 효율적인 서브시퀀스 매칭 방법에 관하여 논의하였다.

타임 워핑 거리는 삼각형 부등식 성질을 만족하지 못한다. 따라서 타임 워핑을 지원하는 기존의 유사 검색 기법에서는 착오 기각의 문제로 인하여 거리 함수를 기반으로 하는 다차원 인덱스를 사용하지 못하였다.

참고 문헌 [16]에서는 인덱스를 기반으로 타임 워핑을 지원하는 효과적인 전체 매칭 기법인 LB_Filter를 제안한 바 있다. 이 기법은 데이터베

이스 내의 각 시퀀스로부터 추출된 특징(feature)들에 대하여 다차원 인덱스를 구축하고, 인덱싱 공간(feature space)에서 사용하기 위한 타임 워핑 거리의 하한 함수인 새로운 거리 함수 D_{tw_lb}를 사용한다. 따라서 타임 워핑 거리 대신 D_{tw_lb}를 필터링 단계에서 사용하더라도 착오 기각이 발생하지 않으며, D_{tw_lb}를 기반으로 하는 인덱스를 사용하더라도 인덱스 검색에서 역시 착오 기각이 발생하지 않는다. 이 결과, 이 기법은 착오 기각 없이 고속의 유사 검색을 지원할 수 있다.

본 논문에서는 LB_Filter를 확장함으로써 타임 워핑을 지원하는 *서브시퀀스 매칭*을 착오 기각 없이 효과적으로 처리하는 기법을 제안하고자 한다. LB_Filer를 단순히 응용하는 방식으로서 *모든 가능한 서브시퀀스들*에 대하여 특징을 추출하고, 이 특징들에 대하여 다차원 인덱스를 구성하는 것을 고려할 수 있다. 그러나 이 방식은 저장 공간 오버헤드 측면에서 큰 무리가 있다.

본 논문에서는 *슬라이딩 윈도우를 기반으로 하는 접두어-질의 기법(prefix-querying method based on sliding windows)*을 제안한다. 제안된 기법에서는 슬라이딩 윈도우(sliding window)[10] 개념을 이용한 인덱싱 전략을 사용한다. 즉, 데이터베이스 내의 각 시퀀스를 고정된 길이의 슬라이딩 윈도우들로 분할하고, 각 윈도우로부터 참고 문헌 [16]의 방식으로 네 개의 특징들을 추출한다. 또한, 이 네 개의 특징들을 인덱싱 애트리뷰트로 이용하여 전체 데이터베이스를 위한 사차원 인덱스를 구성한다. 서브시퀀스 매칭 시에는 질의 시퀀스의 가능한 접두어에 대하여 특징들을 추출하고, 이 특징들과 허용치 ε을 가지고 다차원 인덱스를 검색한다. 인덱스 검색 결과로 반환된 후보들에 대하여 착오 채택 여부를 검사함으로써 최종 질의 결과를 구한다.

제안된 기법의 견고성(robustness)를 증명하기 위하여 제안된 기법을 이용한 서브시퀀스 매칭에서 착오 기각이 발생되지 않음을 증명하였다. 또한, 실제 데이터 및 생성 데이터를 이용한 다양한 실험을 통하여 제안된 기법의 우수성을 규명하였다. 실험 결과로 나타난 전체적인 성능상의 경향을 요약하면 다음과 같다.

(1) TW_Subseq_Matching1과 TW_Subseq_Matching2의 검색 성능을 비교하면, 유사 허용치가 작은 경우에는 TW_Subseq_Matching1이, 그리고 유사 허용치가 큰 경우에는 TW_Subseq_Matching2이 좋은 성능을 보인다.

(2) 제안된 TW_Subseq_Matching1과 TW_Subseq_Matching2는 기존의 LB_Scan 및 ST_Filter와 비교하여 데이터 및 질의 특성에 따라 최대 42배까지의 검색 성능이 개선된다.

(3) minQLen, avgQLen, maxWarpRatio, 데이터 시퀀스들의 수, 데이터 시퀀스들의 평균 길이 등의 인자들이 다양한 값을 갖는 경우에도 제안된 기법은 좋은 성능 개선 효과를 갖는다.

참 고 문 헌

[1] R. Agrawal, C. Faloutsos, and A. Swami, "Efficient Similarity Search in Sequence Databases," In Proc. Int'l. Conference 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 Databases," In Proc. Int'l. Conference 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] 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.

[5] 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.

[6] J. Bercken, B. Seeger, and P. Widmayer, "A General Approach to Bulk Loading Multidimensional Index Structures," In Proc. Int'l. Conf. on Very Large Data Bases, VDLB, pp. 406-415, 1997.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] C. Faloutsos and K. I. Lin, "FastMap: A Fast Algorithm for Indexing, Data-Mining and Visualization of Traditional and Multimedia Datasets," In Proc. Int'l. Conf. on Management of Data, ACM SIGMOD, pp. 163-174, 1995.

[12] 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.

[13] A. Guttman, "R-Trees: A Dynamic Index Structure for Spatial Searching," In Proc. Int'l. Conf. on Mangement of Data, ACM SIGMOD, pp. 47-57, 1984.

[14] I. Kamel and C. Faloutsos, "On Packing R-trees," In Proc. Int'l. Conf. on Information and Knowledge Management, ACM CIKM, pp. 490-499, 1993.

[15] I. Kamel and C. Faloutsos, "Parallel R-trees," In Proc. Int'l. Conf. on Mangement of Data, ACM SIGMOD, pp. 195-204, 1992.

[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. Intl. Conf. on Data Engineering, IEEE, pp. 607-614, 2001.

[17] L. Rabiner and H. H. Juang, Fundamentals of Speech Recognition, Prentice Hall, 1993.

[18] S. T. Leutenegger, J. M. Edgington, and M. A. Lopez, "STR: A Simple and Efficient Algorithm for R-Tree Packing," In Proc. Int'l. Conf. on Data Engineering, IEEE, pp. 497-506, 1997.

[19] 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, Jan. 2001.

[20] 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. Intl. Conf. on Information and Knowledge Management, ACM CIKM, 2000.

[21] 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, pp. 23-32, 2000.

[22] F. P. Preparata and M. Shamos, Computational Geometry: An Introduction, Springer-Verlag, 1985

[23] 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.

[24] T. K. Sellis, N. Roussopoulos, and C. Faloutsos, "The R+-Tree: A Dynamic Index for Multi-Dimensional Objects," In Proc. Int'l. Conf. on Very Large Data Bases, VLDB, 507-518, 1987.

[25] K. S. Shim, R. Srikant, and R. Agrawal, "High-dimensional Similarity Joins," In Proc. Int'l. Conf. on Data Engineering, IEEE, pp. 301-311, Apr. 1997.

[26] G. A. Stephen, String Searching Algorithms, World Scientific Publishing, 1994.

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