시계열 데이타베이스에서 인덱스 보간법을 기반으로 정규화 변환을 지원하는 서브시퀀스 매칭 알고리즘

(An Index Interpolation-based Subsequence Matching Algorithm supporting Normalization Transform in Time-Series Databases)

노 웅 기 [†] 김 상 욱 [††] 황 규 영 [†††]
(Woong-Kee Loh) (Sang-Wook Kim) (Kyu-Young Whang)

요 약 본 논문에서는 시계열 데이타베이스에서 정규화 변환을 지원하는 서브시퀀스 매칭 알고리즘을 제안한다. 정규화 변환은 시계열 데이타 간의 절대적인 유클리드 거리에 관계 없이, 구성하는 값들의 상대적인 변화 추이가 유사한 패턴을 갖는 시계열 데이타를 검색하는 데에 유용하다. 기존의 서브시퀀스 매칭 알고리즘을 확장 없이 정규화 변환 서브시퀀스 매칭에 단순히 응용할 경우, 질의 결과로 반환되어야 할 서브시퀀스를 모두 찾아내지 못하는 착오 기각이 발생한다. 또한, 정규화 변환을 지원하는 기존의 전체 매칭 알고리즘의 경우, 모든 가능한 질의 시퀀스 길이 각각에 대하여 하나씩의 인덱스를 생성하여야 하므로, 저장 공간 및 데이타 시퀀스 삽입/삭제의 부담이 매우 심각하다. 본 논문에서는 인덱스 보간법을 이용하여 문제를 해결한다. **인덱스 보간법**은 인덱스가 요구되는 모든 경우 중에서 적당한 간격의 일부에 대해서만 생성된 인덱스를 이용하며, 인덱스가 필요한 모든 경우에 대한 탐색을 수행하는 기법이다. 제안된 알고리즘은 몇 개의 질의 시퀀스 길이에 대해서만 각각 인덱스를 생성한 후, 이를 이용하여 모든 가능한 길이의 질의 시퀀스에 대해서 탐색을 수행한다. 이때, 착오 기각이 발생하지 않음을 증명한다. 제안된 알고리즘은 질의 시에 주어진 질의 시퀀스의 길이에 따라 생성되어 있는 인덱스 중에서 가장 적절한 것을 선택하여 탐색을 수행한다. 이때, 생성되어 있는 인덱스의 개수가 많을수록 탐색 성능이 향상된다. 필요에 따라 인덱스의 개수를 변화함으로써 탐색 성능과 저장 공간 간의 비율을 유연하게 조정할 수 있다. 질의 시퀀스의 길이 256 ~ 512 중 다섯 개의 길이에 대해 인덱스를 생성하여 실험한 결과, 탐색 결과 선택률이 10^{-2}일 때 제안된 알고리즘의 탐색 성능이 순차 검색에 비하여 평균 2.40 배, 선택률이 10^{-5}일 때 평균 14.6 배 개선되었다. 제안된 알고리즘의 탐색 성능은 탐색 결과 선택률이 작아질수록 더욱 향상되므로, 실제 데이타베이스 응용에서의 효용성이 높다고 판단된다.

Abstract In this paper, we propose a subsequence matching algorithm that supports normalization transform in time-series databases. Normalization transform enables finding sequences with similar patterns even though they are not close to each other in terms of the Euclidean distance. Simple application of the existing subsequence matching algorithms fails to support normalization transform since the algorithms cause false dismissal; i.e., miss part of the final search result. Moreover, the application of the existing whole matching algorithm supporting normalization transform to the subsequence matching requires an index for every possible length of the query sequence, which causes serious overhead on both storage space and time when inserting and deleting data sequences. We

· 본 연구는 첨단정보기술연구센터를 통하여 한국과학재단의 지원을 받았음
† 학생회원 : 한국과학기술원 전자전산학과 전산학전공
woong@mozart.kaist.ac.kr
†† 종신회원 : 강원대학교 정보통신공학부 교수
wook@cc.kangwon.ac.kr
††† 종신회원 : 한국과학기술원 전자전산학과 교수
첨단정보기술연구센터 소장
kywhang@mozart.kaist.ac.kr
논문접수 : 2000년 2월 24일
심사완료 : 2001년 2월 14일

tackle the problem using the notion of index interpolation. *Index interpolation* is a searching method that uses one or more indexes generated for a few selected cases and performs searching for all cases. The proposed algorithm generates indexes only for a small number of different lengths of query sequences. For subsequence matching it selects the most appropriate index among them. We can obtain the better search performance by using more indexes. We formally prove that the proposed algorithm does not cause false dismissal. We can trade-off the search performance with storage space by adjusting the number of indexes. For performance evaluation, we conducted a series of experiments using the indexes for only five different lengths out of lengths 256 ~ 512 of the query sequence. The results show that the proposed algorithm outperforms the sequential scan up to 2.4 times on average when the selectivity of the query is 10^{-2} and up to 14.6 times when it is 10^{-5}. Since the proposed algorithm performs better with smaller selectivities, it is suitable for practical situations, where the queries with smaller selectivities are much more frequent.

1. 서 론

시계열 데이타(time-series data)는 일정한 시간 주기마다 얻어진 연속된 실수 값들로 이루어진 데이타이며, 주가 및 환율 데이타, 날씨 정보, 제품 판매 데이타, 의료 측정 데이타 등의 예가 있다[2, 9]. 시계열 데이타는 데이타 마이닝(data mining), 데이타 웨어하우징(data warehousing) 등의 새로운 데이타베이스 분야에서 점차 중요성이 더해가고 있으며, 시계열 데이타 간의 유사성 문제는 그러한 분야에서 가장 관심을 끄는 문제 중의 하나이다 [2, 3]. 시계열 데이타 간의 유사성 문제의 예로는 주가가 유사하게 변동하는 주식 종목의 검색, 최근의 특정 기간과 유사한 기온 패턴을 갖는 과거 기간의 검색, 특정 제품과 유사한 판매 경향을 보이는 제품의 검색 등을 들 수 있다[9, 19]. 시계열 데이타베이스에 저장된 시계열 데이타를 **데이타 시퀀스**(*data sequence*)라고 부르며, 질의 시퀀스(query sequence)와 유사한 데이타 시퀀스를 검색하는 연산을 **유사 시퀀스 매칭**(*similar sequence matching*)이라고 한다[1, 2, 3, 9, 19].

기존의 유사 시퀀스 매칭 알고리즘은 크게 전체 매칭(whole matching)과 서브시퀀스 매칭(subsequence matching) 알고리즘으로 구분한다[9]. 전체 매칭 알고리즘은 데이타베이스에 저장된 일정한 길이의 데이타 시퀀스들 중에서 동일한 길이의 질의 시퀀스와 유사한 것을 탐색하는 알고리즘이다. 서브시퀀스 매칭 알고리즘은 데이타베이스로부터 임의 길이의 질의 시퀀스와 유사한 서브시퀀스를 포함하는 데이타 시퀀스를 탐색하는 알고리즘이다. 일반적으로, 서브시퀀스 매칭은 전체 매칭에 비하여 보다 다양한 분야에서 응용될 수 있다.

기존의 유사 시퀀스 매칭 알고리즘에서는 길이 n인 데이타 시퀀스를 n 차원 공간 상의 한 점에 대응하며, 두 데이타 시퀀스들 간의 유사성을 대응되는 두 점 간의 유클리드 거리(Euclidean distance)로 정의한다[1, 2, 9, 10, 19]. 또한, n 차원 공간 상의 점들을 효율적으로 저장하고 검색하기 위한 방법으로 R-tree[12], R^*-tree [20], R^+-tree[4] 등의 다차원 인덱스(multidimensional index)를 사용한다. 이러한 다차원 인덱스는 데이타의 차원이 증가함에 따라 탐색 성능이 지수적으로 감소하므로[5, 21], 대부분의 알고리즘에서는 탐색 비용을 감소시키기 위하여 n 차원 공간 상의 점들을 f ($< n$) 차원 공간 상의 점들로 변환하여 처리한다. 이러한 저차원 변환을 위하여 이산 푸리에 변환(Discrete Fourier Transform, DFT)[18], 이산 코사인 변환(Discrete Cosine Transform, DCT)[18], Haar 웨이블렛 변환(Haar Wavelet Transform)[11] 등을 응용한다[1, 7, 9, 10, 19].

기존의 유사 시퀀스 매칭 알고리즘은 데이타 시퀀스와 질의 시퀀스를 비교하기 전에 수행하는 전처리(pre-processing) 변환에 따라 구분할 수도 있다. 참고문헌[1, 9]에서는 데이타 시퀀스와 질의 시퀀스를 전처리 변환 없이 비교하는 알고리즘을 제안하였고, 참고문헌[2, 8, 10, 19, 22]에서는 데이타 시퀀스와 질의 시퀀스에 배수화(scaling), 평행이동(shifting), 정규화(normalization), 이동평균(moving average), 시간왜곡(time warping) 등의 변환을 수행한 결과를 비교하는 알고리즘을 제안하였다. 이러한 전처리 변환의 목적은 응용 분야에 부합되도록 시퀀스 간의 유사성 정의에 유연성을 주는 것이다.

본 논문에서는 정규화 변환[10, 19]을 지원하는 서브시퀀스 매칭 알고리즘에 관하여 논의한다. 정규화 변환을 지원하는 유사 시퀀스 매칭은 데이타 시퀀스 간의 절대적인 유클리드 거리에 관계없이, 구성하는 값들의 상대적인 변화 추이가 유사한 패턴을 갖는 데이타 시퀀스들을 탐색한다[10, 19]. 예를 들어, 절대적인 가격에

상관없이 주가가 상승/하강하는 패턴이 유사한 주식 종목의 검색에 유용하다. 문제를 해결하기 위하여 고려할 수 있는 방법 중의 하나는 기존의 서브시퀀스 매칭 알고리즘[2, 9]을 단순히 응용하는 것이다. 그러나, 이러한 방법은 질의 처리 결과에 포함되어야 할 서브시퀀스를 모두 찾아내지 못하는 착오 기각(false dismissal)이 발생한다. 본 논문에서는 제 3 절에서 이러한 착오 기각이 발생하는 원인을 구체적으로 설명한다. 문제 해결을 위하여 고려할 수 있는 다른 방법은 정규화 변환을 지원하는 기존의 전체 매칭 알고리즘[10]을 서브시퀀스 매칭에 그대로 적용하는 것이다. 전체 매칭 알고리즘이 고정 길이의 질의 시퀀스만을 지원하므로, 임의 길이의 질의 시퀀스를 처리하기 위해서는 모든 가능한 질의 시퀀스 길이 각각에 대하여 하나씩의 다차원 인덱스를 생성해야 한다. 따라서, 인덱스 저장 공간과 데이타 시퀀스 삽입 및 삭제의 부담(overhead)이 매우 심각하다.

본 논문에서는 이러한 문제점들을 극복할 수 있도록 인덱스 보간법[23]을 기반으로 하는 효율적인 정규화 변환 서브시퀀스 매칭 알고리즘을 제안한다. **인덱스 보간법**(*index interpolation*)은 인덱스가 요구되는 모든 경우 중에서 적당한 간격의 일부에 대해서만 생성된 인덱스를 이용하며, 인덱스가 필요한 모든 경우에 대한 탐색을 수행하는 기법이다[23]. 제안된 알고리즘에서는 모든 가능한 질의 시퀀스 길이 중에서 적당한 간격으로 일부에 대해서만 인덱스를 생성하고, 임의 길이의 질의 시퀀스에 대하여 적절한 인덱스를 선택하여 정규화 변환 서브시퀀스 매칭을 수행한다. 제안된 알고리즘의 견고성(robustness)을 규명하기 위하여 질의 처리 결과에서 착오 기각이 발생하지 않음을 증명한다. 인덱스 보간법에서는 필요에 따라 사용하는 인덱스의 개수를 자유롭게 정할 수 있다. 즉, 높은 탐색 성능이 요구되는 경우에는 인덱스의 개수를 증가시키고, 반대로 저장 공간의 절감이 요구되는 경우에는 인덱스의 개수를 감소시킴으로써, 탐색 성능과 저장 공간 간의 비율을 유연하게 조정할 수 있다.

본 논문의 구성은 다음과 같다. 먼저, 제 2 절에서는 정규화 변환을 지원하는 서브시퀀스 매칭 문제를 공식적으로 정의한다. 제 3 절에서는 기존의 서브시퀀스 매칭 및 정규화 변환을 지원하는 유사 시퀀스 매칭 알고리즘을 간략히 설명하고, 정규화 변환 서브시퀀스 매칭에 응용함에 있어서 발생하는 문제점을 지적한다. 제 4 절에서는 정규화 변환을 지원하는 인덱싱 및 탐색 알고리즘을 구체적으로 설명하고, 제 5 절에서는 실험을 통하여 제안된 알고리즘의 성능을 평가한다. 마지막으로, 제 6 절에서는 논문의 내용을 요약하고, 결론을 맺는다.

2. 문제 정의

본 절에서는 정규화 변환 및 정규화 변환을 지원하는 서브시퀀스 매칭 문제를 공식적으로 정의한다. 표 1은 본 논문에서 사용되는 표기법(notation)을 정리한 것이다.

정의 1. 길이 n (≥ 1)인 시퀀스 $\vec{X}=(x_i)$ ($0 \leq i < n$)를 정규화 변환한 시퀀스 $\nu(\vec{X})=(\tilde{x_i})$는 다음과 같이 정의한다[10, 19]:

표 1 표기법 정리

표기법	정 의
$\vec{S}=(s_i)$	하나의 데이타 시퀀스. $\vec{S}=(s_0, \ldots, s_{N-1})$ $(0 \leq i < N)$
$\vec{X}=(x_i)$	데이타 시퀀스 $\vec{S}$에 포함되는 임의의 서브시퀀스. $\vec{X}=(x_0, \ldots, x_{n-1})$ $(0 \leq i < n \leq N)$
$\vec{T}=(t_i)$	질의 시퀀스. $\vec{T}=(t_0, \ldots, t_{n-1})$ $(0 \leq i < n)$
$d(\vec{X}, \vec{T})$	두 시퀀스 $\vec{X}$, $\vec{T}$ 간의 유클리드 거리 $(Len(\vec{X})=Len(\vec{T}))$ $d(\vec{X}, \vec{T})=\{\sum(x_i-t_i)^2\}^{1/2}$
$\vec{X}[s, f]$	시퀀스 $\vec{X}$ 내의 $x_s, \ldots, x_f$ 값으로 구성한 윈도우 $(0 \leq s \leq f < n)$
$\vec{Z}=a\vec{X}+b$	시퀀스 $\vec{X}$의 각 값에 a를 곱하고 b를 더해 생성한 시퀀스. $\vec{Z}=(z_i)=(ax_i+b)$
ε	탐색 영역

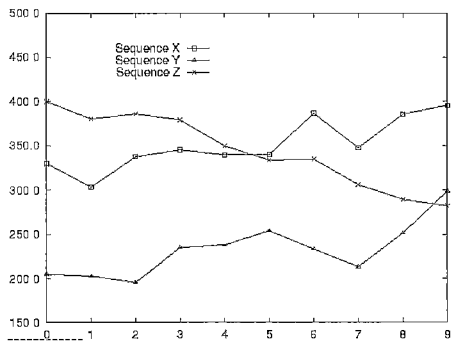
$$\tilde{x}_i = \frac{x_i - \mu(\vec{X})}{\sigma(\vec{X})}$$

여기에서, $\mu(\vec{X})$와 $\sigma(\vec{X})$는 각각 시퀀스 $\vec{X}$의 평균(mean)과 표준 편차(standard deviation)이다. □

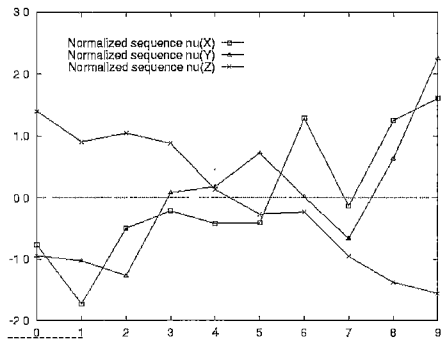
그림 1은 정규화 변환의 예를 보인 것으로, 그림 1(a)는 시퀀스 $\vec{X}$, $\vec{Y}$, $\vec{Z}$에 대한 정규화 변환 전, 그림 1(b)는 정규화 변환 후의 결과를 보인 것이다. 유클리드 거리를 유사성의 척도로 사용할 때, 정규화 변환 전에는 그림 1(a)에서와 같이 시퀀스 $\vec{X}$와 $\vec{Z}$가 유사하다고 판정하나, 정규화 변환 후에는 그림 1(b)에서와 같이 시퀀스 $\nu(\vec{X})$와 $\nu(\vec{Y})$가 유사하다고 판정한다. 즉, 정규화 변환은 데이타 시퀀스를 구성하는 값들이 변화하는 패턴을 비교하는 데에 유용하다.

정규화 변환을 지원하는 서브시퀀스 매칭 문제는 다음과 같이 정의한다. 질의 시에 질의 시퀀스 $\vec{T}$와 탐색 영역 ε이 주어지면, 정규화 변환한 질의 시퀀스 $\nu(\vec{T})$를 비교 대상으로 삼는다. 데이타베이스에 저장된 데이타 시퀀스 $\vec{S}$ 내에 질의 시퀀스 $\vec{T}$와 같은 길이를 갖는 임의의 서브시퀀스 $\vec{X}$를 정규화 변환한 시퀀스 $\nu(\vec{X})$가 $\nu(\vec{T})$와 유사하면, 즉 다음의 공식 (1)을 만족하면, 데이타 시퀀스 $\vec{S}$와 $\vec{S}$ 내에서의 서브시퀀스 $\vec{X}$의 위치를 반환한다.

$$d(\nu(\vec{X}), \nu(\vec{T})) \le \varepsilon \quad (1)$$



(a) 정규화 변환 전



(b) 정규화 변환 후

그림 1 정규화 변환 전과 후의 시퀀스의 예

3. 관련 연구

본 절에서는 기존의 서브시퀀스 매칭 알고리즘[9]과 정규화 변환을 지원하는 기존의 유사 시퀀스 매칭 알고리즘[2, 10]에 대하여 설명한다. 또한, 이러한 알고리즘들을 정규화 변환 서브시퀀스 매칭에 응용할 때 발생하는 문제점들을 지적한다. 제 3.3 절에서는 인덱스 보간법을 이용하는 기존의 알고리즘[23]과 본 논문에서 제안된 알고리즘의 공통점과 차이점을 설명한다.

3.1 서브시퀀스 매칭 알고리즘

참고문헌[9]에서는 참고문헌[1]의 전체 매칭 알고리즘을 확장하여 서브시퀀스 매칭 알고리즘을 제안하였다. 이 알고리즘은 임의의 길이 n의 질의 시퀀스 $\vec{T}$와 탐색 영역 ε을 이용하여 질의를 처리한다. 데이타베이스에 저장된 길이 n의 서브시퀀스 $\vec{X}$와 질의 시퀀스 $\vec{T}$를 비교하기 위하여 그림 2에서와 같이 p 개의 고정된 길이 w의 윈도우(window) $\vec{\chi}_0, \ldots, \vec{\chi}_{(p-1)}$과 $\vec{\tau}_0, \ldots, \vec{\tau}_{(p-1)}$으로 분할한다 ($n = pw$). 참고문헌[9]에서는 서브시퀀스 $\vec{X}$와 질의 시퀀스 $\vec{T}$ 간의 거리가 ε 이내이면, $\vec{\chi}_i$, $\vec{\tau}_i$ ($0 \le i < p$) 쌍 중에서 다음의 공식 (2)를 만족하는 쌍이 반드시 존재함을 보였다.

$$d(\vec{\chi}_i, \vec{\tau}_i) \le \frac{\varepsilon}{\sqrt{p}} \quad (2)$$

또한, 다차원 인덱스를 이용하여 공식 (2)를 만족하는 윈도우 $\vec{\chi}_i$를 빠르게 찾아냄으로써, 최종 질의 결과로 반환될 데이타 시퀀스를 효율적으로 검색하는 알고리즘을

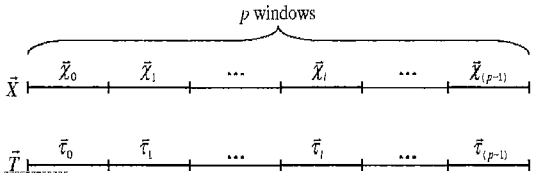


그림 2 서브시퀀스 $\vec{X}$와 질의 시퀀스 $\vec{T}$를 p 개의 윈도우로 분할

제시하였다.

참고문헌[9]의 알고리즘은 아무런 전처리 변환을 지원하지 않으며, 정규화 변환 서브시퀀스 매칭 문제에 단순히 적용할 때 질의 결과로 반환되어야 할 서브시퀀스를 모두 찾아내지 못하는 착오 기각이 발생한다. 이러한 착오 기각이 발생하는 근본적인 원인은 다차원 인덱스

내에 독립적으로 저장된 각각의 윈도우로부터 그 윈도우를 포함하는 서브시퀀스를 정규화 변환하기 위한 정보를 구할 수 없기 때문이다. 참고문헌[9]의 알고리즘은 그림 3과 같이 데이타 시퀀스 $\vec{S}$를 일정한 길이 w의 슬라이딩 윈도우(sliding window) $\vec{\chi}_i$를 단위로 하여 인덱싱 및 탐색을 수행한다. 이때, 슬라이딩 윈도우 $\vec{\chi}_i$를 포함하는 길이 n ($\geq w$)의 서브시퀀스 $\vec{X}$의 개수는 $(n - w + 1)$ 개이며, 각각의 서브시퀀스 $\vec{X}$마다 서로 다른 평균과 표준 편차 값을 갖는다. 그러나, 인덱스에 저장된 $\vec{\chi}_i$에는 이러한 정보가 전혀 포함되어 있지 않으며, 이러한 정보를 포함하기 위한 단순한 방법도 존재하지 않는다. 따라서, 정규화 변환을 위한 정보를 포함하고 있지 않은 슬라이딩 윈도우의 인덱스를 이용하여 정규화 변환 서브시퀀스 매칭을 어떠한 방법으로 수행하더라도 필연적으로 탐색 결과에 착오 기각이 발생하게 된다.

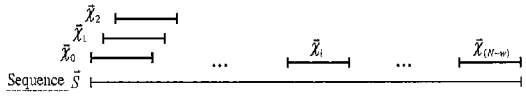


그림 3 데이타 시퀀스 $\vec{S}$의 슬라이딩 윈도우 $\vec{\chi}_i$ 구성

3.2 정규화 변환 유사 시퀀스 매칭 알고리즘

참고문헌[2, 10]에서는 정규화 변환을 지원하는 유사 시퀀스 매칭 알고리즘을 다루었다. 참고문헌[2]에서는 유사한 정규화 변환된 서브시퀀스를 포함하는 모든 데이타 시퀀스 쌍을 검색하는 알고리즘을 제안하였다. 이 알고리즘에서는 질의 시에 탐색 영역 ε과 잡음(noise)의 한계 γ가 주어진다. 본 절에서는 그림 4의 예를 사용하여 참고문헌[2]의 알고리즘을 설명한다. 길의 w의 윈도우 $\vec{\chi}_{1j}$와 $\vec{\chi}_{2j}$ (j = 1, 2, 3)는 각각 데이타 시퀀스 $\vec{S}_1$과 $\vec{S}_2$에 포함되며, 대응되는 윈도우 쌍에 대하여 $d(\nu(\vec{\chi}_{1j}), \nu(\vec{\chi}_{2j})) \leq \varepsilon$을 만족한다. 윈도우 $\vec{\chi}_{11}$, $\vec{\chi}_{12}$ 간의 간격과 윈도우 $\vec{\chi}_{21}$, $\vec{\chi}_{22}$ 간의 간격이 γ 이하이므로

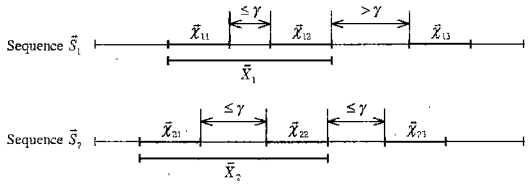


그림 4 참고문헌[2]의 서브시퀀스 매칭

잡음으로 간주한다. 따라서, 두 윈도우를 연결하여 얻어진 서브시퀀스 $\vec{X}_1$과 $\vec{X}_2$를 유사 서브시퀀스로 판정한다. 윈도우 $\vec{\chi}_{12}$, $\vec{\chi}_{13}$ 간의 간격이 γ보다 크므로, 윈도우 $\vec{\chi}_{22}$, $\vec{\chi}_{23}$ 간의 간격이 γ 이하라도, 윈도우를 연결하여 좀더 긴 유사 서브시퀀스를 형성하지 않는다. 만약 $\vec{\chi}_{12}$, $\vec{\chi}_{13}$ 간의 간격이 γ보다 작다면, 윈도우 $\vec{\chi}_{11}$, $\vec{\chi}_{12}$, $\vec{\chi}_{13}$를 연결하여 서브시퀀스 $\vec{X}_1'$을 형성하고, 윈도우 $\vec{\chi}_{21}$, $\vec{\chi}_{22}$, $\vec{\chi}_{23}$를 연결하여 서브시퀀스 $\vec{X}_2'$을 형성한 다음, $\vec{X}_1'$과 $\vec{X}_2'$을 유사 서브시퀀스로 판정한다. 시퀀스 $\vec{X}$에서 추출한 길이 w의 모든 윈도우 $\vec{\chi}_{ij}$에 대하여 정규화 변환한 윈도우 $\nu(\vec{\chi}_{ij})$를 다차원 인덱스에 저장하며, 이 인덱스를 이용하여 유사한 윈도우 $\nu(\vec{\chi}_{ij})$쌍을 빠르게 찾아냄으로써, 최종 질의 결과로 반환될 데이타 시퀀스 쌍을 효율적으로 검색한다.

본 논문에서 풀고자 하는 정규화 변환 서브시퀀스 매칭 문제는 전체 서브시퀀스를 정규화 변환하여 비교하는 반면, 참고문헌[2]의 알고리즘은 서브시퀀스를 구성하는 윈도우를 정규화 변환하여 비교하는 것이다. 즉, 본 논문에서 제안된 알고리즘에서는 주어진 탐색 영역 ε을 전체 서브시퀀스를 비교하기 위한 한도 값으로 사용하는 반면, 참고문헌[2]의 알고리즘에서는 각 윈도우를 비교하기 위한 값으로만 사용한다. 참고문헌[2]의 알고리즘에서는 전체 서브시퀀스를 비교하기 위한 탐색 영역 값은 별도로 존재하지 않는다. 참고문헌[2]의 알고리즘은 일정한 길이의 윈도우 간의 패턴만을 비교하는 반면, 본 논문에서의 정규화 변환 서브시퀀스 매칭 문제는 윈도우를 포함하는 전체 서브시퀀스에 대하여 패턴을 비교하며 임의 길이의 서브시퀀스를 지원한다는 장점을 갖는다. 참고문헌[2]의 알고리즘을 정규화 변환 서브시퀀스 매칭 문제에 적용할 때에도 착오 기각이 발생한다.

그 원인은 제 3.1 절에서 설명한 참고문헌[9]의 알고리즘과 마찬가지로 인덱스에 저장된 슬라이딩 윈도우

$\vec{\chi}_i$에 $\vec{\chi}_i$를 포함하는 길이 n ($\geq w$)의 서브시퀀스 $\vec{X}$에 대한 정규화 변환을 위한 정보가 전혀 포함되어 있지 않기 때문이다. 따라서, 참고문헌[2]의 알고리즘을 어떠한 방법으로 적용하여 정규화 변환 서브시퀀스 매칭을 수행하더라도 필연적으로 탐색 결과에 착오 기각이 발생하게 된다.

참고문헌[10]에서는 길이 n의 두 시퀀스 $\vec{X}$와 $\vec{T}$에 대하여 다음의 공식 (3)을 만족하면 두 시퀀스가 유사하다고 정의하였으며, 주어진 질의 시퀀스 $\vec{T}$와 탐색 영역 ε에 대하여 공식 (3)을 만족하는 데이타 시퀀스 $\vec{X}$에 대한 효율적인 탐색 알고리즘을 제안하였다:

$$\exists a, b(\in R), d(\vec{T}, a\vec{X}+b) \leq \varepsilon \quad (3)$$

여기에서, R은 모든 실수의 집합이다. 데이타 시퀀스 $\vec{X}$에 대하여 공식 (3)의 유사성 조건이 만족되는지 판정하기 위하여 임의의 실수 a, b를 대입해 보는 방법은 매우 비효율적인 방법이다. 따라서, 참고문헌[10]에서는 정의 1에 의하여 정규화 변환된 시퀀스 $\nu(\vec{X})$, $\nu(\vec{T})$에 대해 다음의 공식 (4)를 만족하도록 새로운 탐색 영역 ε'을 제시하였다.

$$\exists a, b(\in R), d(\vec{T}, a\vec{X}+b) \leq \varepsilon \Rightarrow d(\nu(\vec{T}), \nu(\vec{X})) \leq \varepsilon' \quad (4)$$

공식 (4)의 결론부(consequent)의 식을 이용하여 탐색을 수행하여 반환된 데이타 시퀀스 $\vec{X}$의 집합이 전제부(antecedent)의 식을 만족하는 $\vec{X}$의 집합을 포함하므로 착오 기각이 발생하지 않는다. 참고문헌[10]의 알고리즘은 모든 데이타 시퀀스 $\vec{X}$에 대하여 정규화 변환한 시퀀스 $\nu(\vec{X})$를 다차원 인덱스에 저장하고, 이를 이용하여 질의 시퀀스 $\nu(\vec{T})$로부터 탐색 영역 ε' 이내의 시퀀스 $\nu(\vec{X})$를 효율적으로 탐색한다.

참고문헌[10]에서는 데이타 시퀀스와 질의 시퀀스의 길이가 일정한 전체 매칭만을 다루었으며 서브시퀀스 매칭으로의 확장에 대해서는 언급하지 않았다. 참고문헌[10]의 알고리즘을 단순히 적용하여 임의 길이의 서브시퀀스 매칭을 수행할 때, 다음과 같은 문제가 발생한다. 일반적으로, 길이 n의 임의의 시퀀스 $\vec{X}$, $\vec{T}$와 윈도우 $\overrightarrow{X[s,f]}$, $\overrightarrow{T[s,f]}$ ($0 \leq s \leq f < n$)에 대하여 다음의 공식 (5)가 성립한다:

$$d(\vec{X}, \vec{T}) \leq \varepsilon \Rightarrow d(\overrightarrow{X[s,f]}, \overrightarrow{T[s,f]}) \leq \varepsilon \quad (5)$$

그러나, 정규화 변환된 시퀀스들에 대해서는 위의 공식 (5)가 성립하지 않는다. 즉,

$$d(\nu(\vec{X}), \nu(\vec{T})) \leq \varepsilon \not\Rightarrow d(\nu(\overrightarrow{X[s,f]}), \nu(\overrightarrow{T[s,f]})) \leq \varepsilon \quad (6)$$

따라서, 공식 (6)의 결론부를 만족하는 데이타 시퀀스 $\vec{X}$의 집합이 전제부의 식을 만족하는 $\vec{X}$의 집합을 포함하지 않는다. 즉, 질의 시퀀스의 길이 $n_0 = f - s + 1$에 대하여 생성된 인덱스를 이용하여 길이 $n(> n_0)$의 질의 시퀀스에 대한 정규화 변환 서브시퀀스 매칭을 수행하면 착오 기각이 발생하게 된다.

3.3 인덱스 보간법에 기반한 기존의 알고리즘

인덱스 보간법에 기반하여 서브시퀀스 매칭을 수행하는 기존의 알고리즘은 참고문헌[23]에서 제안되었다. 참고문헌[23]의 알고리즘과 본 논문에서 제안하는 알고리즘은 인덱스 보간법을 이용한다는 공통점이 있는 반면, 동시에 다음과 같은 세가지 차이점을 갖는다. (1) 인덱스 보간법은 일부 인덱스만으로 모든 경우에 사용할 수 있다는 원칙일 뿐이며, 어떤 조건에 따라 어떻게 인덱스 보간법을 적용할 것인지는 알고리즘에 따라 다르다. 즉, 참고문헌[23]에서는 서로 다른 몇 개의 이동평균 계수(moving average order)에 대하여 인덱스를 생성하였으며, 본 논문에서는 서로 다른 몇 개의 질의 시퀀스 길이에 따라 정규화 변환을 위한 인덱스를 생성하였다. (2) 이러한 인덱스를 이용하여 탐색 결과로 착오 기각이 발생하지 않음을 보이기 위한 증명 과정과 구체적인 탐색 방법도 완전히 다르다. 구체적인 내용은 제 4 절에서 설명한다. (3) 참고문헌[23]의 알고리즘은 본 논문에서의 문제를 해결하기 위하여 사용될 수 없다. 그 이유는 이동평균 변환과 정규화 변환의 정의는 전혀 다르며, 정규화 변환에는 이동평균 계수라는 개념이 존재하지 않기 때문이다.

4. 정규화 변환 서브시퀀스 매칭 기법

본 절에서는 정규화 변환을 지원하는 서브시퀀스 매칭을 위한 인덱싱 및 탐색 알고리즘을 제안한다. 제 4.1 절에서는 먼저 문제 해결을 위한 기본 아이디어를 설명하고, 제 4.2 절에서는 인덱싱 및 탐색 알고리즘을 구체적으로 설명한다.

4.1 기본 아이디어

인덱스 보간법은 인덱스가 요구되는 모든 경우 중에서 일부에 대해서만 인덱스를 생성하여 탐색을 수행하는 방법이므로, 인덱스 보간법을 적용하기 위하여 가장 중요한 점은 인덱스가 생성되어 있지 않은 경우에 대하여 다른 인덱스를 선택하여 사용할 때, 착오 기각이 발생하지 않음을 보이는 것이다. 예를 들어, 길의 n의 질의 시퀀스에 대하여 길이 w ($< n$)에 대한 인덱스를 이용하여 탐색을 수행할 때, 최종적으로 반환되어야 할 서브시퀀스를 모두 찾아내지 못하는 현상이 발생하지 않

아야 한다는 점이다. 본 절에서는 이러한 착오 기각이 발생하지 않도록 인덱스 보간법을 적용하는 방법에 대하여 설명한다.

본 논문에서는 참고문헌[10]의 알고리즘을 확장하여 문제를 해결한다. 참고문헌[10]의 알고리즘을 확장 없이 단순히 응용하여 정규화 변환 서브시퀀스 매칭을 수행하기 위한 방법은 모든 가능한 질의 시퀀스 길이에 대하여 하나씩의 인덱스를 생성하는 것이다. 그 이유는 제 3.2 절에서 설명한 바와 같이 하나의 정해진 길이 n_0에 대하여 생성된 인덱스를 이용하여 임의 길이 $n(>n_0)$의 질의 시퀀스에 대한 정규화 변환 서브시퀀스 매칭을 수행할 때 착오 기각이 발생하기 때문이다. 그러나, 이러한 방법은 저장 공간 및 데이타 시퀀스 삽입/삭제의 부담이 매우 심각하다.

본 논문에서는 이러한 문제를 극복하기 위하여, 미리 선택된 몇 개의 질의 시퀀스 길이 w에 대해서만 인덱스를 생성하고, 이를 이용하여 임의 길이 n ($\geq w$)의 질의 시퀀스에 대한 탐색을 수행하는 새로운 탐색 기법을 제안한다. 본 논문에서는 길이 w의 질의 시퀀스에 대하여 생성된 인덱스를 **w-인덱스**(w-*index*)라 정의한다. 주어진 질의 시퀀스의 길이 n에 대하여, 인덱스를 생성한 질의 시퀀스 길이 w 중에 같은 값이 있으면 그 값에 대한 w-인덱스를 이용하고, 같은 값이 없으면 w-인덱스 중에서 다음의 공식 (7)에 의하여 하나를 선택하여 탐색을 수행한다.

$$\omega = \max\{w \mid w < n\} \tag{7}$$

이때, 공식 (7)에 의해 선택되어 질의 처리에 사용되는 인덱스를 **ω-인덱스**(ω-*index*)라 정의한다.

질의 시퀀스의 길이 n이 ω 값과 같지 않은 경우의 탐색을 위하여 다음의 정리 1이 필요하다. 정리 1은 탐색 영역 ε을 대치하며 탐색 결과에 착오 기각이 발생하지 않음을 보장하는 새로운 탐색 영역 ε'을 제시한다[1]. 정리 1의 도입을 위하여 본 논문에서 제안하는 알고리즘에서는 시퀀스를 구성하는 모든 값들에 -1을 곱하는 역전(reverse) 변환을 동시에 지원하지 않는다. 즉, 다음의 공식 (8)을 만족하는 서브시퀀스 $\vec{X}$는 최종 탐색 결과로 반환하지 않는다.

$$d(-\nu(\vec{X}), \nu(\vec{T})) \leq \varepsilon \tag{8}$$

만약 데이타베이스 응용에서 공식 (8)을 만족하는 서브시퀀스를 최종 탐색 결과에 포함하도록 요구하는 경우에는, 먼저 질의 시퀀스를 역전 변환하여 이를 대상으로 제안된 알고리즘을 수행하면 된다.

정리 1. 길이 n (≥ 1)인 임의의 시퀀스 $\vec{X}=(x_i)$, $\vec{T}=(t_i)$ ($0 \leq i < n$)에 대하여 다음이 성립한다 ($0 \leq s \leq f < n$):

$$d(\nu(\vec{X}), \nu(\vec{T})) \leq \varepsilon \Rightarrow d(\nu(\overrightarrow{X[s,f]}), \nu(\overrightarrow{T[s,f]})) \leq \varepsilon' \tag{9}$$

여기에서, ε'은 다음과 같으며, ω는 윈도우 $\overrightarrow{T[s,f]}$의 길이이다 ($\omega = f-s+1$).

$$\varepsilon' = \sqrt{2\omega - 2\sqrt{\omega^2 - \omega \cdot \varepsilon^2 \cdot \frac{\sigma^2(\vec{T})}{\sigma^2(\overrightarrow{T[s,f]})}}} \tag{10}$$

증명: 부록 참조. □

공식 (10)에서 안쪽의 제곱근 내의 값이 0 이상이어야 한다. 즉, ε'을 계산함에 있어서 다음 공식 (11)의 조건을 검토하여야 한다. 공식 (10)에서 바깥쪽의 제곱근 내의 값은 항상 0 이상이다.

$$\omega > \varepsilon^2 \cdot \frac{\sigma^2(\vec{T})}{\sigma^2(\overrightarrow{T[s,f]})} \tag{11}$$

질의 시퀀스의 길이 n이 공식 (7)에 의하여 얻어진 ω와 같지 않으면, ω-인덱스를 이용하여 공식 (9)의 결론부의 식인 다음의 공식 (12)를 만족하는 모든 서브시퀀스 $\vec{X}$로 후보 집합을 구성한다:

$$d(\nu(\overrightarrow{X[s,f]}), \nu(\overrightarrow{T[s,f]})) \leq \varepsilon' \tag{12}$$

윈도우 $\overrightarrow{T[s,f]}$의 길이가 ω이므로, 공식 (12)를 이용한 탐색을 수행하기 위하여 질의 시퀀스 $\vec{T}$로부터 추출 가능한 윈도우 $\overrightarrow{T[s,f]}$의 개수는 $(n-\omega+1)$ 개다. 이들 중에서 새로운 탐색 영역 ε'을 이용한 탐색의 성능을 최대화하기 위하여 다음의 공식 (13)에 의하여 윈도우 $\overrightarrow{T[s,f]}$를 결정한다 ($0 \leq s' \leq f' < n$, $f'-s' = \omega-1$):

$$\overrightarrow{T[s,f]} = \{\overrightarrow{T[s',f']} \mid \sigma(\overrightarrow{T[s',f']}) \text{is maximum.}\} \tag{13}$$

탐색 영역 ε'이 클수록 공식 (12)를 이용한 탐색에 의한 착오 채택(false alarm)의 가능성이 커지므로, 가급적 ε'을 작게 설정하여야 한다. 공식 (10)에서 ε' 값을 변화시킬 수 있는 인자는 $\sigma(\overrightarrow{T[s,f]})$ 뿐이며, $\sigma(\overrightarrow{T[s,f]})$ 값이 최대가 될 때 ε' 값이 최소가 된다. 따라서, 공식 (13)에 의해 결정된 윈도우 $\overrightarrow{T[s,f]}$를 이용하여 탐색을 수행할 때 최고의 탐색 성능을 얻을 수 있다.

탐색 영역 ε'에 대해 ω-인덱스를 이용하여 공식 (12)를 만족하는 탐색을 수행할 때, 정리 1을 통하여 착오 기각이 발생하지 않음을 보인다. 정리 1에서 제시한 공식 (9)의 전제부의 조건을 만족하는 모든 서브시퀀스 $\nu(\vec{X})$에 대하여, 윈도우 $\nu(\overrightarrow{X[s,f]})$가 항상 결론부의 조

1) 탐색 영역 ε'은 질의 시퀀스 $\vec{T}$를 구성하는 값에 따라 ε보다 클 수도 있고 작을 수도 있다.

건을 만족한다. 즉, 공식 (9)의 결론부의 조건을 만족하는 $(\nu(\overrightarrow{X[s,f]}),\nu(\overrightarrow{T[s,f]}))$ 쌍의 집합은 전제부의 조건을 만족하는 대응되는 $(\nu(\vec{X}),\nu(\vec{T}))$ 쌍의 집합을 포함한다. 따라서, 결론부의 식에 의하여 탐색을 수행할 때 착오 기각이 발생하지 않는다.

4.2 인덱싱 및 탐색 알고리즘

본 절에서는 w-인덱스를 생성하는 알고리즘과 w-인덱스를 이용한 탐색 알고리즘에 대하여 설명한다. w-인덱스는 정규화 변환된 윈도우를 저장한다는 점을 제외하면 기본적으로 참고문헌[10]에서와 같은 방법을 이용한다. 길이 N의 데이타 시퀀스 $\vec{S}$로부터 그림 3과 같이 길이 w의 $(N - w + 1)$ 개의 슬라이딩 윈도우 $\vec{\chi}_i = (\chi_{ij}) = \overrightarrow{X[i, i+w-1]}$ $(0 \le i \le N-w,\ 0 \le j < w)$를 추출한다. 각 슬라이딩 윈도우 $\vec{\chi}_i$를 정규화 변환한 윈도우 $\nu(\vec{\chi}_i) = (\tilde{\chi}_{ij})$에 대하여 $(\phi_{i0}, \cdots, \phi_{i(f-1)})$을 키로 f 차원 인덱스를 구성한다. 여기에서, ϕ_{i0}, $\cdots$, $\phi_{i(f-1)}$ 값들은 $\nu(\vec{\chi}_i)$를 구성하는 $\vec{\chi}_{i0}$, $\cdots$, $\vec{\chi}_{i(w-1)}$ 값들에 대하여 DFT 변환을 수행하고 항상 0이 아닌 앞의 $f(<w)$ 개의 계수들만을 선택한 것이다[1, 9, 10]. 제안된 알고리즘은 주어진 질의 시퀀스의 길이에 대하여 w-인덱스가 생성되어 있지 않은 경우에도 탐색을 수행할 수 있으나, 이러한 경우를 위하여 w-인덱스에 추가적으로 저장하는 정보는 없다. 따라서, 참고문헌[10]의 알고리즘과 비교하여 w-인덱스의 크기는 증가하지 않는다.

w-인덱스를 이용한 정규화 변환 서브시퀀스 매칭 알고리즘은 다음과 같다. 주어진 질의 시퀀스의 길이 n이 w-인덱스를 생성한 w 값들 중에 존재하면 참고문헌[10]에서와 같은 탐색을 수행한다. 즉, 질의 시퀀스 길이 n에 대하여 w-인덱스가 생성되어 있다면, 그 w-인덱스를 이용하여 단순히 질의 시퀀스 $\nu(\vec{T})$와 ε 범위 내의 서브시퀀스 $\nu(\vec{X})$를 검색한다. 이때, 서론에서 설명한 바와 같이, 길이 n의 $\nu(\vec{T})$와 $\nu(\vec{X})$는 각각 n-차원의 점들로 매핑되며, 다차원 인덱스를 이용하여 ε 범위의 효율적인 영역 탐색(range search)을 수행한다. 본 절에서는 n이 w 값들 중에 존재하지 않는 경우에 대하여 중점적으로 설명한다.

그림 5의 알고리즘 *NormalizedSearch*($\vec{T}$, ε)은 질의 시퀀스 $\vec{T}$와 탐색 영역 ε을 입력으로 받아 공식 (1)을 만족하는 서브시퀀스 $\vec{X}$를 탐색하는 알고리즘이다. 이 알고리즘은 일반적인 다차원 인덱스 구조를 이용할 수 있다. 제안된 알고리즘은 질의 시퀀스 내의 윈도우 $\overrightarrow{T[s,f]}$와 새로운 탐색 영역 ε'을 이용하여 탐색을 수행한다는 점 이외에 참고문헌[10]의 알고리즘과 다른점이 없으므로, w-인덱스를 이용하여 후보집합을 구하기까지 추가적인 처리 비용이 들지 않는다.

그림 5의 알고리즘의 각 라인을 설명하면 다음과 같다. 먼저, 라인 (1)에서 공식 (7)에 의해 ω를 구하고, 질의 시퀀스 $\vec{T}$로부터 공식 (13)에 따라 길이 ω의 윈도우 $\overrightarrow{T[s,f]}$를 결정한다. 구해진 ω와 $\overrightarrow{T[s,f]}$를 이용하여 라인 (2)에서 공식 (11)이 만족되는지 검사한다. 라인 (3) ~ (5)는 공식 (11)이 만족될 때, 라인 (7)은 만족되지 않을 때의 처리 과정을 나타낸다. 라인 (3)에서 공식 (10)에 따라 새로운 탐색 영역 ε'을 구하고, 라인 (4)에서 질의 윈도우 $\nu(\overrightarrow{T[s,f]})$와 탐색 영역 ε'에 대하여 w-인덱스를 이용하여 영역 탐색을 수행한다. 라인 (5)에서 탐색의 결과로 반환된 모든 윈도우 $\nu(\overrightarrow{X[s,f]})$로 후보 집합을 구성한다. 라인 (7)에서는 공식 (11)이 만족되지 않으므로 데이타베이스 내의 모든 가능한 윈도우 $\nu(\overrightarrow{X[s,f]})$로 후보 집합을 구성한다. 라인 (9) ~ (12)는 후보 집합에 포함된 각 윈도우에 대하여 최종 탐색 결과로서의 적합성을 판정하는 과정이다. 라인 (10)에서 윈도우 $\overrightarrow{X[s,f]}$로부터 구해진 길의 n의 서브시퀀스 $\vec{X}$를 직접 디스크로부터 읽는다. 라인 (11)에서

```
Procedure NormalizedSearch(Sequence T, Range ε)
// Passed parameters
Sequence T;      // query sequence
Range ε;         // search range
(1) Find ω and T[s,f];          // using Eq. (7) and (13)
(2) if Eq. (11) is satisfied then
(3)     Compute ε';             // using Eq. (10)
(4)     Perform range search using ν(T[s,f]) and ε';
(5)     Make a candidate set C consisting of the
        returned windows;
(6) else
(7)     Make a candidate set C consisting of all the
        windows in database;
(8) endif
(9) for each window ν(X[s,f]) in C do
(10)    Read the corresponding subsequence X from
        disk;
(11)    if d(ν(X),ν(T))≤ε then return X;
(12) end for
```

그림 5 정규화 변환 서브시퀀스 매칭 알고리즘

정규화 변환된 두 시퀀스 $\nu(\vec{X})$와 $\nu(\vec{Y})$ 간의 유클리드 거리가 ε 이내이면 서브시퀀스 $\vec{X}$를 최종 결과로 반환한다.

5. 성능 평가

본 절에서는 실험을 통하여 제안된 알고리즘의 성능을 평가한다. 본 실험의 가장 중요한 목적은 몇 개의 w-인덱스만을 이용하더라도 모든 질의 시퀀스 길이 n에 대하여 인덱스가 생성되어 있는 경우에 비해 제안된 알고리즘의 탐색 성능이 크게 저하되지 않음을 보이는 것이다. 이에 보충하여, 본 실험에서는 선택률이 감소하거나 사용하는 인덱스의 개수가 증가함에 따라 탐색 성능이 향상되며, 제안된 알고리즘이 순차 검색에 비하여 탐색 성능이 우수함을 보인다. 모든 질의 시퀀스 길이 n에 대하여 인덱스가 생성되어 있는 경우에 수행하는 알고리즘은 참고문헌[10]의 알고리즘을 확장 없이 단순히 응용한 것이다. 제 5.1 절에서는 실험 환경 설정에 대하여, 제 5.2 절에서는 실험 결과 및 분석에 대하여 서술한다.

5.1 실험 환경

본 실험에 사용한 데이타베이스는 1994년 11월부터 1998년 5월까지 길이 1024의 한국 주가 데이타 620 개 종목의 데이타 시퀀스로 구성된다. 본 논문에서는 랜덤 데이타(random data)를 이용한 실험 결과를 포함하지 않으며, 그 이유는 랜덤 데이타를 이용한 실험 결과가 실제의 데이타를 이용한 실험 결과와 거의 유사하기 때문이다[16, 24]. 또한, 기존의 연구들에서도 실제의 데이타만을 사용하거나[2], 랜덤 데이타를 사용하더라도 실제의 데이타를 사용한 실험 결과와 유사하게 나타났기 때문이다[9, 19, 22]. 제안된 알고리즘의 실험을 위하여 선택된 질의 시퀀스 길이 w = 256, 320, 384, 448, 512에 대하여 w-인덱스를 생성하였고, 모든 질의 시퀀스 길이에 대하여 인덱스를 생성하여 참고문헌[10]의 알고리즘을 확장 없이 응용하는 경우의 실험을 위하여 질의 시퀀스 길이 n = 256 + 32i (i = 0, ... , 8)인 경우와 n = w ± 1인 경우에 대하여 인덱스를 생성하였다. 인덱스를 n = w ± 1인 경우에 대하여 생성하는 것은 질의 시퀀스의 길이가 n = w - 1에서 n = w + 1로 변할 때 탐색 성능이 어떻게 변화하는지 관찰하기 위함이다. 본 실험에서는 인덱스의 차원을 줄이기 위하여 DFT 변환을 사용하였다. 인덱스 차원을 여러 가지로 변화하며 실험하여 가장 탐색 성능이 우수한 f = 6 차원을 선택하였다. 실험에서 사용된 인덱스의 평균 크기는 데이타베이스 크기에 비하여 약 43.5 %를 나타냈다. 따라서, 인덱스에 의한 저장 공간의 부담을 줄이는 것이 더욱 중요하다고 할 수 있다. 질의 시퀀스를 생성하기 위하여 먼저 참고문헌[9]에서와 같이 임의의 128 개 종목의 데이타 시퀀스들로부터 임의 위치에서 길이 n의 서브시퀀스 $Q=(q_i)$ ($0 \le i < n$)를 추출하였고, 이를 이용하여 참고문헌[1]에서와 같이 다음의 공식 (14)에 따라 질의 시퀀스 $T=(t_i)$ ($0 \le i < n$)를 생성하였다.

$$t_i = q_i + z_i,\ z_i \in (-50, 50) \tag{14}$$

여기에서, z_i는 (-50, 50) 범위 내의 임의의 값이며, 50은 모든 $\vec{Q}$에 대하여 $|q_{i+1} - q_i|$ ($0 \le i < n - 1$) 값들을 평균한 값의 약 5 %이다. 탐색 영역 ε은 128 개의 질의 시퀀스 각각에 대하여 다음의 공식 (15)에 보인 선택률(selectivity)을 기준으로 결정하였고, 실험에 사용된 선택률 값들은 0.00001, 0.0001, 0.001, 0.01이다.

$$\text{선택률} = \frac{\text{탐색 결과 반환된 서브시퀀스의 개수}}{\text{모든 가능한 데이타 서브시퀀스의 개수}} \tag{15}$$

본 실험에서는 다차원 인덱스 구조로 R*-tree[4]를 이용하였으며, 데이타 시퀀스로부터 추출한 슬라이딩 윈도우는 참고문헌[9]에서와 같이 여러 개의 슬라이딩 윈도우를 하나의 서브트레일(subtrail) 형태로 저장하였다. 본 실험을 수행한 하드웨어 플랫폼은 인텔 Celeron 400MHz CPU, 128MB RAM, 2.0GB 하드 디스크를 장착한 PC이며, 소프트웨어 플랫폼은 마이크로소프트 한글 윈도우 NT 4.0 운영 체제(operating system, OS)이다.

5.2 실험 결과 및 분석

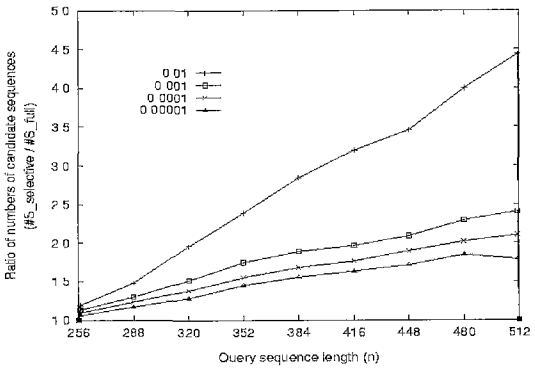
본 실험에서 각 데이타 시퀀스는 하나의 블록(block)에 저장하며, 모든 블록은 디스크에 순차적으로 저장하였다. 이러한 저장 구조를 이용하여 다음과 같이 제안된 알고리즘을 수행하였다. 먼저, 인덱스를 통하여 후보 서브시퀀스 집합을 구한다. 후보 서브시퀀스들을 그들이 포함된 데이타 시퀀스 별로 그룹화하고, 디스크에 블록이 저장된 순서로 정렬(sort)한다. 후보 서브시퀀스가 포함된 데이타 시퀀스를 하나하나 디스크로부터 읽어들인다. 각 데이타 시퀀스로부터 후보 서브시퀀스를 추출하여 정규화 변환하고 질의 시퀀스와의 거리를 구한다. 거리가 ε 이하인 경우 해당 서브시퀀스를 최종 결과로 반환한다. 하나의 데이타 시퀀스를 디스크로부터 읽는 시간을 t_1, 후보 서브시퀀스들이 포함된 데이타 시퀀스의 개수를 #CS라 하면, 아무런 가정이 없는 경우 모든 데이타 시퀀스를 읽는 시간은 대략 $t_1 \cdot$ #CS가 된

다. 그러나, 본 실험에서와 같이 모든 블록이 디스크에 순차적으로 저장된 경우에는, 새로운 블록을 읽기 위하여 디스크 헤드(head)가 임의의 방향이 아닌 한 방향으로만 이동하므로, 디스크 헤드가 이동하는 시간이 절약된다. 따라서, 모든 데이타 시퀀스를 읽는 시간은 $t_l \cdot \#CS$보다 훨씬 짧아진다.

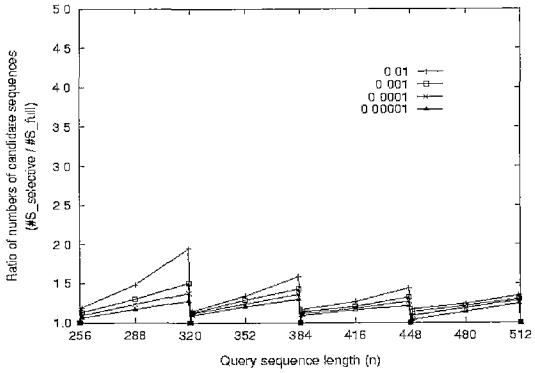
첫번째 실험은 주어진 질의 시퀀스의 길이 n에 대한 w-인덱스가 생성되어 있지 않은 경우와 생성되어 있는 경우에 탐색 알고리즘을 실행하여 얻어진 후보집합 내의 서브시퀀스 개수를 비교하는 실험이다. 이 실험의 목적은 질의 시퀀스 길이 n에 대한 w-인덱스가 생성되어 있지 않은 경우에 제안된 알고리즘에 의하여 얼마만큼의 착오 채택이 더 발생하는지 비교하기 위한 것이다.

그림 6은 첫번째 실험에 의한 결과를 보인 것이다. 그림 6(a)는 w = 256, 512인 두 개의 w-인덱스만을 이용한 결과이고, 그림 6(b)는 w = 256, 320, 384, 448, 512인 다섯 개의 w-인덱스를 이용한 결과이다. 가로 축은 질의 시퀀스의 길이 n을 나타내고, 세로 축은 인덱스가 없는 경우의 후보 서브시퀀스 개수 $\#S_{selective}$를 인덱스가 있는 경우의 후보 서브시퀀스 개수 $\#S_{full}$로 나눈 비율 값을 나타낸다. 이 값은 128 개의 질의 시퀀스에 대하여 얻어진 후보 서브시퀀스 개수 비율 값들을 평균한 것이다. 실험 결과, 그림 6(a)에서 질의 시퀀스의 길이가 511일 때, 선택률이 10^{-2}인 경우 후보 서브시퀀스 개수 비율이 약 4.43 배까지, 선택률이 10^{-5}인 경우 약 1.79 배까지 증가하였다. 그림 6(b)에서는 질의 시퀀스의 길이가 319일 때, 선택률이 10^{-2}인 경우 후보 서브시퀀스 개수 비율이 약 1.95 배까지, 선택률이 10^{-5}인 경우 약 1.28 배까지 증가하였다. 질의 시퀀스의 길이가 511일 때에는, 선택률이 10^{-2}과 10^{-5}인 경우 각각에 대하여 약 1.34 배와 1.24 배까지 증가하였다. 다섯 개의 w-인덱스를 사용할 때 모든 질의 시퀀스 길이에 대해 인덱스가 생성되어 있는 경우와 비교하여 제안된 알고리즘의 착오 채택 비율이 크게 증가하지 않았다. 인덱스의 개수는 257 개 중에서 다섯 개만을 사용하여 51.4 (= 257 / 5) 배의 이득(gain)을 얻은 반면, 착오 채택 비율은 최대 약 1.95 배까지만 증가하였다. 또한, 선택률이 감소할수록, 많은 개수의 w-인덱스를 사용할수록 후보 서브시퀀스 개수 비율이 감소하였다.

그림 6에서 질의 시퀀스 길이 n이 w-인덱스를 생성한 길이 w로부터 멀어짐에 따라 후보 서브시퀀스 개수 비율이 증가하는 것으로 나타났다. 그 원인은, $\frac{n}{w} = \frac{Len(\vec{T})}{Len(T[s,f])}$ 값이 증가함에 따라 제 4.1 절에서 주어진 공식 (10)의 $\frac{\sigma(\vec{T})}{\sigma(T[s,f])}$ 값이 대체로 증가하게 되어 새로운 탐색 영역 ε' 값이 점점 커지기 때문이다. 그림 6에서 선택률이 증가하는 경우에도 후보 서브시퀀스 개수 비율이 증가하는 것으로 나타났다. 그 원인은, 선택률이 증가하면 $\frac{\varepsilon'}{\varepsilon}$의 비율이 점점 커지기 때문이다. 그림 6(b)에서 제안된 알고리즘의 탐색에 사용하는 다섯 개의 w-인덱스의 w 값이 커짐에 따라 후보 서브시퀀스 개수 비율이 대체로 감소하는 것으로 나타났다. 그 원인은, w 값이 커짐에 따라 공식 (10)의 $\frac{\sigma(\vec{T})}{\sigma(T[s,f])}$ 값이 감소하게 되어 새로운 탐색 영역 ε' 값도 마찬가지로 감소하기 때문이다.



(a) 두 개의 w-인덱스 이용



(b) 다섯 개의 w-인덱스 이용

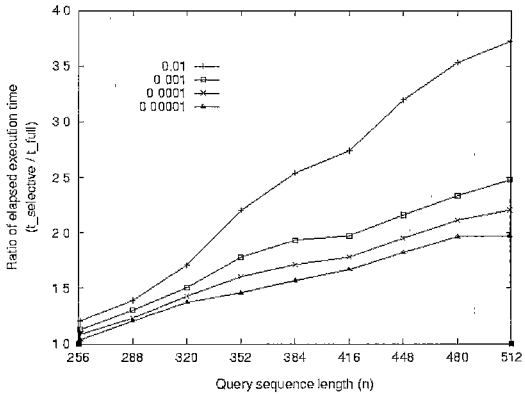
그림 6 후보 집합 내의 서브시퀀스 개수 비율 ($\#S_{selective}$ / $\#S_{full}$).

두번째 실험은 주어진 질의 시퀀스의 길이 n에 대한 w-인덱스가 생성되어 있지 않은 경우와 생성되어 있는 경우의 탐색 알고리즘의 실행 시간(elapsed execution time)을 비교하는 실험이다. 이 실험의 목적은 질의 시퀀스 길이 n에 대한 w-인덱스가 생성되어 있지 않은 경우에 제안된 알고리즘의 실행 시간이 얼마나 더 걸리는지 비교하기 위한 것이다. 데이타베이스 프로그램의 전체 실행 시간은 CPU 작업 시간과 디스크 액세스 시간의 합이다. 이 실험에서는 실제의 디스크 I/O를 보장하기 위하여 OS에 의한 버퍼링(buffering)을 수행하지 않는 루틴들을 사용하였다[13].

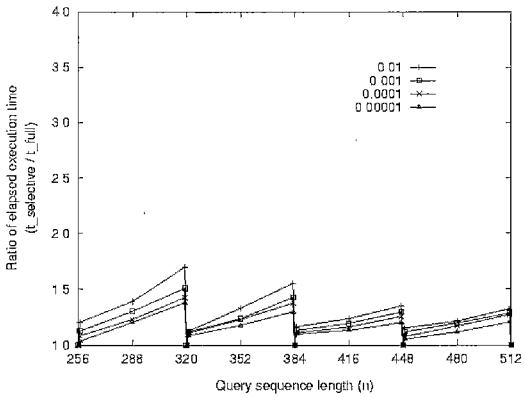
그림 7은 두번째 실험에 의한 결과를 보인 것이며, 그림 7(a)는 w = 256, 512인 두 개의 w-인덱스만을 이용한 결과이고, 그림 7(b)는 w = 256, 320, 384, 448, 512인 다섯 개의 w-인덱스를 이용한 결과이다. 세로 축은 인덱스가 없는 경우의 실행 시간 $t_{selective}$를 인덱스가 있는 경우의 실행 시간 t_{full}로 나눈 비율 값을 나타낸다. 이 값은 첫번째 실험에서와 마찬가지로 128 개의 질의 시퀀스에 대하여 얻어진 실행 시간 비율 값들을 평균한 것이다.

실험 결과, 실행 시간 비율도 그림 6의 후보 서브시퀀스 개수 비율과 대체로 유사한 경향을 나타내었다. 그림 7(a)에서 질의 시퀀스의 길이가 511일 때, 선택률이 10^{-2}인 경우 실행 시간 비율이 약 3.72 배까지, 선택률이 10^{-5}인 경우 약 1.97 배까지 증가하였다. 그림 7(b)에서는 질의 시퀀스의 길이가 319일 때, 선택률이 10^{-2}인 경우 실행 시간 비율이 약 1.70 배까지, 선택률이 10^{-5}인 경우 약 1.38 배까지 증가하였다. 질의 시퀀스의 길이가 511일 때에는, 선택률이 10^{-2}과 10^{-5}인 경우 각각에 대하여 약 1.33 배와 1.21 배까지 증가하였다. 인덱스의 개수에 있어서의 이득은 51.4 배에 달한 반면, 탐색 시간 비율은 최대 약 1.70 배까지만 증가하였다. 또한, 첫번째 실험에서와 같이 선택률이 감소할수록, 많은 개수의 w-인덱스를 사용할수록 실행 시간 비율이 감소하였다.

세번째 실험은 제안된 알고리즘과 순차 검색 알고리즘의 실행 시간을 비교하는 실험이다. 이 실험의 목적은 질의 시퀀스 길이 n에 대한 w-인덱스가 생성되어 있지 않은 경우에 제안된 알고리즘이 순차 검색에 비하여 얼마나 성능이 개선되는가를 보이기 위한 것이다. 이 실험에서도 두번째 실험과 마찬가지로 OS에 의한 버퍼링을 수행하지 않는 루틴들을 사용하였다[13].



(a) 두 개의 w-인덱스 이용



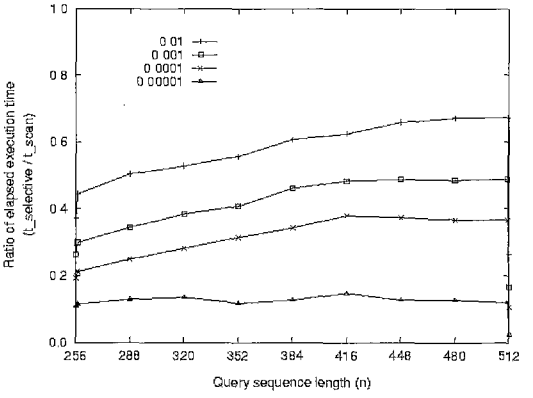
(b) 다섯 개의 w-인덱스 이용

그림 7 실행 시간 비율 ($t_{selective}$ / t_{full}).

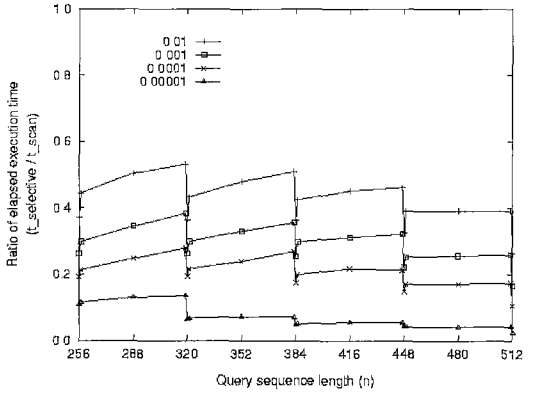
순차 검색은 다음과 같이 수행하였다. 먼저, 각 데이타 시퀀스를 하나하나 순서대로 디스크로부터 읽어 들인다. 각 데이타 시퀀스로부터 질의 시퀀스와 같은 길이의 모든 가능한 서브시퀀스를 추출한다. 이때, 데이타 시퀀스의 길이가 N, 질의 시퀀스의 길이가 n이면, 가능한 모든 서브시퀀스의 개수는 $N - n + 1$ 개다. 각 서브시퀀스를 정규화 변환하여 질의 시퀀스와의 거리를 구하고, 거리가 ε 이하인 경우 해당 서브시퀀스를 최종 결과로 반환한다. 하나의 데이타 시퀀스를 디스크로부터 읽는 시간을 t_1, 전체 데이타 시퀀스의 개수를 $\#DS$라 하면, 아무런 가정이 없는 경우 전체 데이타 시퀀스를 읽는 시간은 대략 $t_1 \cdot \#DS$가 된다. 그러나, 본 실험에서와 같이 모든 블록이 디스크에 순차적으로 저장된 경우에는, 제안된 알고리즘에서와 같이 디스크 헤드가 이

동하는 시간이 절약되므로, 전체 데이타 시퀀스를 읽는 시간은 $t_1 \cdot \#DS$보다 훨씬 짧아진다.

그림 8은 세번째 실험에 의한 결과를 보인 것이며, 그림 8(a)는 w = 256, 512인 두 개의 w-인덱스만을 이용한 결과이고, 그림 8(b)는 w = 256, 320, 384, 448, 512인 다섯 개의 w-인덱스를 이용한 결과이다. 세로 축은 제안된 알고리즘이 w-인덱스를 이용하여 탐색에 걸린 실행 시간 $t_{selective}$를 순차 검색 알고리즘의 실행 시간 t_{scan}으로 나눈 비율 값을 나타낸다. 이 값은 128 개의 질의 시퀀스에 대하여 얻어진 실행 시간 비율 값들을 평균한 것이다. 실험 결과, 그림 8(a)에서 두 개의 w-인덱스를 사용할 때 제안된 알고리즘의 탐색 성능이 순차 검색에 비하여 개선된 배율을 선택률 별로 평균하면, 선택률 $10^{-2} \sim 10^{-5}$에 대하여 각각 약 1.87 (1/0.54), 2.58 (1/0.39), 3.46 (1/0.29), 8.61 (1/0.12) 배로 나타났다. 그림 8(b)에서 다섯 개의 w-인덱스를 사용할 때의 배율을 선택률 별로 평균하면, 선택률 $10^{-2} \sim 10^{-5}$에 대하여 각각 약 2.40 (1/0.42), 3.49 (1/0.29), 4.97 (1/0.20), 14.6 (1/0.07) 배로 나타났다. 또한, 앞에서의 실험에서와 마찬가지로 선택률이 감소할수록, 많은 개수의 w-인덱스를 사용할수록 탐색 성능이 향상되었다.



(a) 두 개의 w-인덱스 이용



(b) 다섯 개의 w-인덱스 이용

그림 8 실행 시간 비율 ($t_{selective}$ / t_{scan})

6. 요약 및 결론

본 논문에서는 정규화 변환을 지원하는 효율적인 서브시퀀스 매칭에 대하여 논의하였다. 기존의 서브시퀀스 매칭 알고리즘[2, 9]을 정규화 변환 서브시퀀스 매칭에 단순히 적용하면, 질의 결과로 반환되어야 할 서브시퀀스를 모두 찾아내지 못하는 착오 기각이 발생한다. 또한, 정규화 변환을 지원하는 전체 매칭 알고리즘[10]을 서브시퀀스 매칭에 적용하면, 모든 가능한 질의 시퀀스 길이 각각에 대하여 하나씩의 인덱스를 생성하여야 하므로, 저장 공간 및 데이타 시퀀스 삽입/삭제의 부담이 매우 심각하다.

본 논문에서는 인덱스 보간법[23]을 기반으로 참고문헌[10]의 알고리즘을 확장하여 문제를 해결하였다. 제안된 알고리즘은 미리 적당한 간격으로 선택한 몇 개의 질의 시퀀스 길이 w에 대해서만 w-인덱스를 생성하며, 이를 이용하여 임의의 질의 시퀀스 길이 n에 대하여 정규화 변환 서브시퀀스 매칭을 수행한다. 이때, 제안된 알고리즘을 이용한 정규화 변환 서브시퀀스 매칭에서 착오 기각이 발생하지 않음을 증명하였다.

제안된 알고리즘의 성능을 검증하기 위하여 다양한 실험을 수행하였다. 실험 결과, 사용하는 인덱스의 개수가 증가할수록 더 나은 탐색 성능을 얻을 수 있으며, 모든 질의 시퀀스 길이 n에 대해 인덱스가 생성되어 있는 경우와 비교하여 제안된 알고리즘의 탐색 성능이 크게 저하되지 않는 것으로 나타났다. 질의 시퀀스의 길이 256 ~ 512 중 다섯 개의 w-인덱스를 사용하여 순차 검색과 탐색 성능을 비교한 결과, 제안된 알고리즘의 성능은 선택률이 10^{-2}인 경우 평균 2.4 배, 선택률이 10^{-5}인 경우 평균 14.6 배 개선되었다. 응용 분야에 따라 탐색에 이용하는 w-인덱스의 개수를 조정함으로써 더욱 개선된 탐색 성능을 얻을 수 있다.

대부분의 데이타베이스 응용에서는 탐색 결과 선택률이 작은 질의가 선택률이 큰 질의보다 훨씬 빈번하다. 제안된 알고리즘은 탐색 결과 선택률이 작아질수록 탐색 성능이 더욱 향상되므로, 실제 데이타베이스 응용 환

경에서 제안된 알고리즘의 실질적인 효용성이 높다고 판단된다.

참 고 문 헌

[1] Agrawal, R. et al., "Efficient Similarity Search in Sequence Databases," In *Proc. Int'l Conf. on Foundations of Data Organization and Algorithms*, pp. 69-84, Chicago, Illinois, Oct. 1993.

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

[3] Agrawal, R. et al., "Querying Shapes of Histories," In *Proc. Int'l Conf. on Very Large Data Bases*, pp. 502-514, Zurich, Switzerland, Sept. 1995.

[4] Beckmann, N. 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, Atlantic City, NJ, June 1990.

[5] Berchtold, S. et al., "The X-tree: An Index Structure for High-Dimensional Data," In *Proc. Int'l Conf. on Very Large Data Bases*, pp. 28-39, Mumbai, India, Sept. 1996.

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

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

[8] Chu, K. K. W. and Wong, M. H., "Fast Time-Series Searching with Scaling and Shifting," In *Proc. ACM SIGACT-SIGMODSIGART Symposium on Principles of Data- base Systems*, pp. 237-248, Philadelphia, Pe- nnsylvania, May 1999.

[9] Faloutsos, C. et al., "Fast Subsequence Matching in Time-Series Databases," In *Proc. Int'l Conf. on Management of Data*, ACM SIGMOD, pp.419-429, Minneapolis, Minnesota, June 1994.

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

[11] Gonzalez, R. C. and Woods, R. E., *Digital Image Processing*, Addison-Wesley, 1993.

[12] Guttman, A., "R-trees: A Dynamic Index Structure for Spatial Searching," In *Proc. Int'l Conf. on Management of Data*, ACM SIGMOD, pp. 47-57, Boston, Massachusetts, June 1984.

[13] Hart, J. M., *Win32 System Programming*, Addison-Wesley Developers Press, 1997.

[14] Kendall, M., *Time-Series*, 2nd Ed., Charles Griffin and Company, 1976.

[15] Kreyszig, E., *Advanced Engineering Mathematics*, 7th Ed., John Wiley & Sons, 1993.

[16] Moon, Y.-S. et al., "Efficient Time-Series Subsequence Matching Using Duality in Constructing Windows," *Information Systems*, accepted to appear.

[17] Oppenheim, A. V. and Schafer, R. W., *Digital Signal Processing*, Prentice-Hall, 1975.

[18] Press, W. H. et al., *Numerical Recipes in C - The Art of Scientific Computing*, 2nd Ed., Cambridge University Press, 1992.

[19] Rafiei, D. and Mendelzon, A., "Similarity-Based Queries for Time Series Data," In *Proc. Int'l Conf. on Management of Data*, ACM SIGMOD, pp. 13-25, Tucson, Arizona, June 1997.

[20] Sellis, T. et al., "The R+-tree: A Dynamic Index for Multidimensional Objects," In *Proc. Int'l Conf. on Very Large Data Bases*, pp. 507-518, Brighton, England, Sept. 1987.

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

[22] Yi, B.-K. et al., "Efficient Retrieval of Similar Time Sequences Under Time Warping," In *Proc. Int'l Conf. on Data Engineering*, IEEE, pp. 201-208, Orlando, Florida, Feb. 1998.

[23] 노웅기, 김상욱, 황규영, 심규석, "시계열 데이타베이스에서 임의 계수의 이동평균 변환을 지원하는 서브시퀀스 매칭 알고리즘," **정보과학회 논문지: 데이타베이스**, 제 27 권, 제 9 호, pp. 469-485, 2000년 9월.

[24] 문양세, 노웅기, 황규영, "윈도우 구성의 이원성을 이용한 효율적인 시계열 서브시퀀스 매칭," **정보과학회 논문지: 데이타베이스**, 게재 승인.

부 록

정리 1의 증명: 정리 1은 다음에 기술하는 보조 정리 1, 2, 3, 4와 따름 정리 1을 이용하여 증명한다. 시퀀스 $\vec{X}$, $\vec{T}$가 윈도우 $\vec{\chi}=\overrightarrow{X[s,f]}$, $\vec{\tau}=\overrightarrow{T[s,f]}$ $(0 \le s \le f < n)$를 포함하므로 다음의 부등식이 성립한다.

$$d^2(\nu(\vec{X}),\nu(\vec{T})) = d^2\left(\frac{\vec{X}-\mu(\vec{X})}{\sigma(\vec{X})}, \frac{\vec{T}-\mu(\vec{T})}{\sigma(\vec{T})}\right)$$

$$\ge d^2\left(\frac{\vec{\chi}-\mu(\vec{X})}{\sigma(\vec{X})}, \frac{\vec{\tau}-\mu(\vec{T})}{\sigma(\vec{T})}\right)$$

보조 정리 2와 4를 통하여 다음의 부등식을 얻는다.

$$d^2\left(\frac{\vec{\chi}-\mu(\vec{X})}{\sigma(\vec{X})}, \frac{\vec{\tau}-\mu(\vec{T})}{\sigma(\vec{T})}\right) \geq \frac{\sigma^2(\vec{\tau})}{\sigma^2(\vec{T})}\left(D^2-\frac{D^4}{4\omega}\right)$$

위의 두 부등식을 조합하면 다음의 부등식이 성립한다.

$$d^2(\nu(\vec{X}),\nu(\vec{T})) \geq \frac{\sigma^2(\vec{\tau})}{\sigma^2(\vec{T})}\left(D^2-\frac{D^4}{4\omega}\right)$$

따라서, 다음의 공식을 얻을 수 있다.

$$d^2(\nu(\vec{X}),\nu(\vec{T})) \leq \varepsilon^2 \Rightarrow \frac{\sigma^2(\vec{\tau})}{\sigma^2(\vec{T})}\left(D^2-\frac{D^4}{4\omega}\right) \leq \varepsilon^2 \qquad (16)$$

공식 (16)의 결론부의 식에서 이차 방정식의 근의 공식(equation of roots)을 이용하여 D^2의 범위를 구하면 다음과 같다.

$$D^2 \leq 2\omega - 2\sqrt{\omega^2-\omega\cdot\varepsilon^2\cdot\frac{\sigma^2(\vec{T})}{\sigma^2(\vec{\tau})}} \ \vee$$

$$D^2 \geq 2\omega + 2\sqrt{\omega^2-\omega\cdot\varepsilon^2\cdot\frac{\sigma^2(\vec{T})}{\sigma^2(\vec{\tau})}}$$

그림 9에서 두 시퀀스 $\frac{\vec{\chi}}{\sigma(\vec{X})}$와 $\frac{\vec{\tau}}{\sigma(\vec{T})}$ 사이의 각이 90° 를 넘는 경우는 정규화와 동시에 역전 변환된 시퀀스에 대한 탐색이므로 두 시퀀스 사이의 각을 90° 이하로 한정한다. 이때, 그림 9에서 $D^2 \leq 2\omega$이므로 위의 두 번째 부등식은 버린다. 위의 첫번째 부등식에서 제곱을 없애면 다음과 같다.

$$0 \leq D \leq \sqrt{2\omega - 2\sqrt{\omega^2-\omega\cdot\varepsilon^2\cdot\frac{\sigma^2(\vec{T})}{\sigma^2(\vec{\tau})}}} \qquad (17)$$

따라서, 공식 (9)가 성립한다. □

보조 정리 1. 길이 n ($\geq$ 1)인 임의의 시퀀스 $\vec{X}=(x_i)$ ($0 \leq i < n$)에 대하여 다음의 식을 최소로 하는 δ_1의 값은 $\delta_1=\mu(\vec{X})$이다.

$$D_1(\delta_1)=\sum_{i=0}^{n-1}(x_i-\delta_1)^2$$

여기에서, $\mu(\vec{X})$는 시퀀스 $\vec{X}$의 평균이다.

증명: $D_1(\delta_1)$의 1차 미분과 2차 미분을 구하면,

$$\frac{\partial}{\partial\delta_1}D_1 = 2n\delta_1 - 2\sum_{i=0}^{n-1}x_i$$

$$\frac{\partial^2}{\partial\delta_1^2}D_1 = 2n \ (>0)$$

$D_1(\delta_1)$의 1차 미분을 0으로 하는 δ_1 값을 찾으면,

$$\delta_1 = \sum_{i=0}^{n-1}\frac{x_i}{n} = \mu(\vec{X})$$

이며, $D_1(\delta_1)$의 2차 미분이 항상 0보다 크므로, $D_1(\delta_1)$은 $\delta_1=\mu(\vec{X})$에서 최소가 된다. □

따름 정리 1. 길이 n ($\geq$ 1)인 임의의 시퀀스 $\vec{X}=(x_i)$, $\vec{T}=(t_i)$ ($0 \leq i < n$)와 임의의 상수 δ에 대하여 다음의 식이 성립한다:

$$\sum_{i=0}^{n-1}(x_i-t_i-\delta)^2 \geq \sum_{i=0}^{n-1}(x_i-t_i-(\mu(\vec{X})-\mu(\vec{T})))^2$$

여기에서, $\mu(\vec{X})$, $\mu(\vec{T})$는 시퀀스 $\vec{X}$, $\vec{T}$의 평균이다.

증명: 제시된 부등식의 좌변을 다음과 같이 $D_2(\delta_2)$로 정의한다.

$$D_2(\delta_2)=\sum_{i=0}^{n-1}(x_i-t_i-\delta_2)^2$$

이때, $D_2(\delta_2)$를 최소로 하는 δ_2의 값은 $\delta_2=\mu(\vec{X})-\mu(\vec{T})$임을 보인다.

시퀀스 $\vec{Z}$를 $\vec{Z}=\vec{X}-\vec{T}$라고 정의한다.

즉, $\vec{Z}=(z_i)=(x_i-t_i)$ ($0 \leq i < n$)이다. $D_2(\delta_2)$는 다음과 같이 고쳐 쓸 수 있다.

$$D_2(\delta_2)=\sum_{i=0}^{n-1}(z_i-\delta_2)^2$$

보조 정리 1에 의하여 $D_2(\delta_2)$를 최소로 하는 δ_2의 값은 다음과 같다.

$$\delta_2=\mu(\vec{Z})=\mu(\vec{X}-\vec{T})=\mu(\vec{X})-\mu(\vec{T})$$

따라서, 제시한 부등식이 성립한다. □

보조 정리 2. 길이 n ($\geq$ 1)인 임의의 시퀀스 $\vec{X}=(x_i)$, $\vec{T}=(t_i)$ ($0 \leq i < n$) 내의 동일한 위치의 길이 ω ($1 \leq \omega \leq n$)인 임의의 윈도우 $\vec{\chi}=\overrightarrow{X[s,f]}=(\chi_j)$, $\vec{\tau}=\overrightarrow{T[s,f]}=(\tau_j)$ ($0 \leq s \leq f < n$, $0 \leq j < \omega$)에 대하여, 다음이 성립한다:

$$d^2\left(\frac{\vec{\chi}-\mu(\vec{X})}{\sigma(\vec{X})}, \frac{\vec{\tau}-\mu(\vec{T})}{\sigma(\vec{T})}\right) \geq d^2\left(\frac{\vec{\chi}-\mu(\vec{\chi})}{\sigma(\vec{X})}, \frac{\vec{\tau}-\mu(\vec{\tau})}{\sigma(\vec{T})}\right)$$

여기에서, $\mu(\vec{X})$, $\mu(\vec{T})$, $\sigma(\vec{X})$, $\sigma(\vec{T})$는 각각 시퀀스 $\vec{X}$와 $\vec{T}$의 평균과 표준 편차이고, $\mu(\vec{\chi})$, $\mu(\vec{\tau})$는 윈도우 $\vec{\chi}$와 $\vec{\tau}$의 평균이다.

증명: 제시된 부등식의 좌변을 다음과 같이 고쳐 쓸 수 있다.

$$d^2\left(\frac{\vec{\chi}-\mu(\vec{X})}{\sigma(\vec{X})}, \frac{\vec{\tau}-\mu(\vec{T})}{\sigma(\vec{T})}\right)$$

$$= \sum_{j=0}^{\omega-1}\left(\frac{\chi_j-\mu(\vec{X})}{\sigma(\vec{X})} - \frac{\tau_j-\mu(\vec{T})}{\sigma(\vec{T})}\right)^2$$

$$= \sum_{j=0}^{\omega-1}\left(\left(\frac{\chi_j'}{\sigma(\vec{X})} - \frac{\tau_j'}{\sigma(\vec{T})}\right) - \left(\frac{u}{\sigma(\vec{X})} - \frac{v}{\sigma(\vec{T})}\right)\right)^2$$

여기에서, $\chi_j'=\chi_j-\mu(\vec{\chi})$, $\tau_j'=\tau_j-\mu(\vec{\tau})$이며, $u=\mu(\vec{X})-\mu(\vec{\chi})$, $v=\mu(\vec{T})-\mu(\vec{\tau})$는 상수이다.

윈도우 $\frac{\vec{\chi}'}{\sigma(\vec{X})} = \left(\frac{\chi_j'}{\sigma(\vec{X})}\right)$ $(0 \le j < \omega)$의 평균 $\mu\left(\frac{\vec{\chi}'}{\sigma(\vec{X})}\right)$을 구하면 다음과 같다.

$$\mu\left(\frac{\vec{\chi}'}{\sigma(\vec{X})}\right) = \frac{\mu(\vec{\chi}')}{\sigma(\vec{X})} = \frac{\mu(\vec{\chi}-\mu(\vec{\chi}))}{\sigma(\vec{X})} = \frac{\mu(\vec{\chi})-\mu(\vec{\chi})}{\sigma(\vec{X})} = 0$$

마찬가지로 $\mu\left(\frac{\vec{\tau}'}{\sigma(\vec{T})}\right)=0$을 얻을 수 있다.

따름 정리 1과 위의 결과에 의하여 다음이 성립한다.

$$\sum_{j=0}^{\omega-1}\left(\left(\frac{\chi_j'}{\sigma(\vec{X})}-\frac{\tau_j'}{\sigma(\vec{T})}\right)-\left(\frac{u}{\sigma(\vec{X})}-\frac{v}{\sigma(\vec{T})}\right)\right)^2$$

$$\ge \sum_{j=0}^{\omega-1}\left(\left(\frac{\chi_j'}{\sigma(\vec{X})}-\frac{\tau_j'}{\sigma(\vec{T})}\right)-\left(\mu\left(\frac{\vec{\chi}'}{\sigma(\vec{X})}\right)-\mu\left(\frac{\vec{\tau}'}{\sigma(\vec{T})}\right)\right)\right)^2$$

$$= \sum_{j=0}^{\omega-1}\left(\frac{\chi_j'}{\sigma(\vec{X})}-\frac{\tau_j'}{\sigma(\vec{T})}\right)^2$$

위의 식을 다시 쓰면 다음과 같다.

$$\sum_{j=0}^{\omega-1}\left(\frac{\chi_j-\mu(\vec{X})}{\sigma(\vec{X})}-\frac{\tau_j-\mu(\vec{T})}{\sigma(\vec{T})}\right)^2$$

$$\ge \sum_{j=0}^{\omega-1}\left(\frac{\chi_j-\mu(\vec{\chi})}{\sigma(\vec{X})}-\frac{\tau_j-\mu(\vec{\tau})}{\sigma(\vec{T})}\right)^2 \qquad \square$$

보조 정리 3. 길이 n (≥ 1)인 임의의 시퀀스 $\vec{X}$를 정규화 변환한 시퀀스 $\nu(\vec{X})$에 대하여 다음 식이 성립한다:

$$\|\nu(\vec{X})\| = \sqrt{n}$$

여기에서, $\|\nu(\vec{X})\|$는 $\nu(\vec{X})$의 길이(norm)이다.

증명: 좌변의 제곱을 구하면

$$\|\nu(\vec{X})\|^2 = \|\frac{\vec{X}-\mu(\vec{X})}{\sigma(\vec{X})}\|^2$$

$$= \sum_{i=0}^{n-1}\left(\frac{x_i-\mu(\vec{X})}{\sigma(\vec{X})}\right)^2$$

$$= \frac{n}{\sigma^2(\vec{X})}\sum_{i=0}^{n-1}\frac{(x_i-\mu(\vec{X}))^2}{n}$$

$$= \frac{n}{\sigma^2(\vec{X})}\cdot\sigma^2(\vec{X})$$

$$= n$$

양변에 제곱근을 취하면,

$$\|\nu(\vec{X})\| = \sqrt{n} \qquad \square$$

보조 정리 4. 길이 n (≥ 1)인 임의의 시퀀스 $\vec{X}$, $\vec{T}$ 내의 동일한 위치의 길이 ω $(1 \le \omega \le n)$인 임의의 윈도우 $\vec{\chi}=\overrightarrow{X[s,f]}$, $\vec{\tau}=\overrightarrow{T[s,f]}$에 대하여, 다음 식이 성립한다:

$$d^2\left(\frac{\vec{\chi}-\mu(\vec{\chi})}{\sigma(\vec{X})}, \frac{\vec{\tau}-\mu(\vec{\tau})}{\sigma(\vec{T})}\right) \ge \frac{\sigma^2(\vec{\tau})}{\sigma^2(\vec{T})}\left(D^2-\frac{D^4}{4\omega}\right)$$

여기에서,

$$D = d(\nu(\vec{\chi}), \nu(\vec{\tau}))$$

이며, $\sigma(\vec{X})$, $\sigma(\vec{T})$는 시퀀스 $\vec{X}$, $\vec{T}$의 표준 편차이고, $\mu(\vec{\chi})$, $\mu(\vec{\tau})$, $\sigma(\vec{\chi})$, $\sigma(\vec{\tau})$는 윈도우 $\vec{\chi}$, $\vec{\tau}$의 평균과 표준 편차이다.

증명: 그림 9는 두 윈도우 $\vec{\chi}'=\vec{\chi}-\mu(\vec{\chi})$와 $\vec{\tau}'=\vec{\tau}-\mu(\vec{\tau})$ 간의 공간 상의 거리 관계를 보인 그림이다. 그림 9에서 정규화 변환된 윈도우 $\nu(\vec{\chi})=\frac{\vec{\chi}'}{\sigma(\vec{\chi})}$와 $\nu(\vec{\tau})=\frac{\vec{\tau}'}{\sigma(\vec{\tau})}$는 보조 정리 3에 의하여 모두 $\sqrt{\omega}$의 길이를 가지며, 윈도우 $\frac{\vec{\tau}'}{\sigma(\vec{T})}$는 $\sqrt{\omega}\cdot\frac{\sigma(\vec{\tau})}{\sigma(\vec{T})}$의 길이를 갖는다. 그림 9에서 점 P는 정규화 변환된 윈도우 $\frac{\vec{\chi}'}{\sigma(\vec{\chi})}$의 연장선 상에 원점 O로부터의 거리가 $\sqrt{\omega}\cdot\frac{\sigma(\vec{\tau})}{\sigma(\vec{T})}$인 위치에 잡은 점이므로, 원점 O와 점 P, 윈도우 $\frac{\vec{\tau}'}{\sigma(\vec{T})}$을 잇는 삼각형은 양변의 길이가 $\sqrt{\omega}\cdot\frac{\sigma(\vec{\tau})}{\sigma(\vec{T})}$인 이등변 삼각형이다. 따라서, 다음의 식들이 성립한다 $(a \ge 0,\ b \ge 0,\ c \ge 0)$:

$$a^2+c^2 = \omega\cdot\frac{\sigma^2(\vec{\tau})}{\sigma^2(\vec{T})}$$

$$b^2+c^2 = D^2\cdot\frac{\sigma^2(\vec{\tau})}{\sigma^2(\vec{T})}$$

$$a+b = \sqrt{\omega}\cdot\frac{\sigma(\vec{\tau})}{\sigma(\vec{T})}$$

위의 식들을 c에 대하여 정리하면 다음과 같다:

$$c^2 = \frac{\sigma^2(\vec{\tau})}{\sigma^2(\vec{T})}\left(D^2-\frac{D^4}{4\omega}\right)$$

그림 9에서 윈도우 $\frac{\vec{\chi}'}{\sigma(\vec{X})}$는 정규화 변환된 윈도우 $\nu(\vec{\chi})=\frac{\vec{\chi}'}{\sigma(\vec{\chi})}$의 연장선 상에 위치한다. 이때, 두 윈도우 $\frac{\vec{\chi}'}{\sigma(\vec{X})}$와 $\frac{\vec{\tau}'}{\sigma(\vec{T})}$ 간의 거리 E는 항상 c보다 크거나 같으므로 다음이 성립한다.

$$E^2 = d^2\left(\frac{\vec{\chi}'}{\sigma(\vec{X})}, \frac{\vec{\tau}'}{\sigma(\vec{T})}\right) \ge \frac{\sigma^2(\vec{\tau})}{\sigma^2(\vec{T})}\left(D^2-\frac{D^4}{4\omega}\right) \qquad \square$$

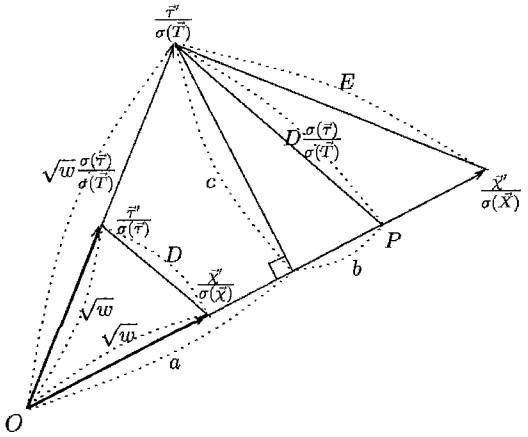


그림 9 두 윈도우 $\vec{\chi}'$, $\vec{\tau}'$ 간의 공간 상의 거리 관계

노 웅 기
정보과학회논문지 : 데이타베이스
제 28 권 제 1 호 참조

김 상 욱
1989년 2월 서울대학교 컴퓨터공학과 학사. 1991년 2월 한국과학기술원 전산학과 석사. 1994년 2월 한국과학기술원 전산학과 박사. 1991년 7월 ~ 8월 미국 Stanford 대학 (캘리포니아 주 소재) 전산학과 방문 (Gio Wiederhold 교수 초청). 1994년 3월 ~ 1995년 2월 한국과학기술원 정보전자연구소 Post-Doc. 1995년 3월 ~ 현재 강원대학교 정보통신공학과 조교수. 1999년 8월 ~ 현재 미국 IBM Watson Research Center 방문 (Philip Yu 박사 초청). 관심 분야는 데이타베이스, 데이타 마이닝, 멀티미디어 내용기반 검색, 주기억장치 DBMS, 공간 DBMS/GIS, 인터넷 데이타베이스

황 규 영
정보과학회 논문지 : 데이타베이스
제 28 권 제 1 호 참조