# 시계열 데이타베이스에서 유사한 서브시퀀스의 모양 기반 검색

## (Shape-Based Retrieval of Similar Subsequences in Time-Series Databases)

윤 지 희 † 김 상 욱 †† 김 태 훈 ††† 박 상 현 ††††

(JeeHee Yoon) (Sang-Wook Kim) (Tae-Hoon Kim) (Sang-Hyun Park)

**요 약** 본 논문에서는 시계열 데이타베이스에서의 모양 기반 검색 문제에 관하여 논의한다. 모양 기반 검색은 실제 요소 값과 관계없이 질의 시퀀스와 유사한 모양을 갖는 (서브)시퀀스를 찾는 연산이다. 본 연구에서는 모양 기반 서브시퀀스 검색을 위한 새로운 기법을 제안한다. 먼저, 시프팅, 스케일링, 이동 평균, 타임 워핑 등 변환들의 다양한 조합을 지원하는 모양 기반 검색을 위하여 새로운 유사 모델을 제시한다. 또한, 이러한 유사 모델을 기반으로 하는 모양 기반 검색을 효과적으로 처리하기 위하여 효율적인 인덱싱 및 질의 처리 기법들을 제안한다. 제안된 기법의 유용성을 규명하기 위하여 실제 데이타인 S&P 500 주식 데이타를 이용한 다양한 실험을 수행한다. 실험 결과에 의하면, 제안된 기법은 질의 시퀀스의 모양과 유사한 모양을 갖는 서브시퀀스들을 성공적으로 검색할 뿐만 아니라 순차 검색 기법과 비교하여 66배까지의 상당한 성능 개선 효과를 갖는 것으로 나타났다.

**키워드** : 시퀀스 데이타베이스, 유사 검색, 모양기반 검색, 인덱싱, 타임 워핑

***Abstract*** This paper deals with the problem of *shape-based retrieval* in time-series databases. The shape-based retrieval is defined as the operation that searches for the (sub)sequences whose shapes are similar to that of a given query sequence regardless of their actual element values. In this paper, we propose an effective and efficient approach for shape-based retrieval of subsequences. We first introduce a new similarity model for shape-based retrieval that supports various combinations of transformations such as shifting, scaling, moving average, and time warping. For efficient processing of the shape-based retrieval based on the similarity model, we also propose the indexing and query processing methods. To verify the superiority of our approach, we perform extensive experiments with the real-world S&P 500 stock data. The results reveal that our approach successfully finds all the subsequences that have the shapes similar to that of the query sequence, and also achieves significant speedup up to around 66 times compared with the sequential scan method.

**Key words** : Sequence Databases, Similarity Search, Shape-Based Retrieval, Indexing, Time Warping

· 본 연구는 한국학술진흥재단 신도연구자 지원사업(과제번호: KRF-2000-041-E00258), 한국과학재단 목적기초 연구사업(과제번호: R05-2002-000-01085-0), 2002년도 한림대학교 지원 학술 연구 조성비 등의 부분적인 지원을 받았습니다.

† 정 회 원 : 한림대학교 정보통신공학부 교수
jhyoon@hallym.ac.kr
†† 종신회원 : 강원대학교 컴퓨터정보통신공학부 교수
*wook@kangwon.ac.kr*
††† 학생회원 : (주)클래러스 연구원
egoist@hallym.ac.kr
†††† 정 회 원 : 포항공과대학교 컴퓨터공학과 교수
sanghyun@postech.ac.kr

논문접수 : 2001년 7월 10일
심사완료 : 2002년 5월 28일

## 1. 서 론

시계열 데이타베이스(time-series database)란 객체의 변화되는 값들의 연속으로 구성된 데이타 시퀀스(data sequence: 이후부터 간략히 시퀀스라 칭함)들의 집합이다[1]. 대표적인 예로는 주가 데이타, 환율 데이타, 기온 데이타, 제품 판매량 데이타, 기업 성장률 데이타 등이 있다[2][3]. 유사 검색(similarity search)이란 주어진 질의 시퀀스(query sequence)와 변화의 패턴이 유사한 시퀀스들을 시퀀스 데이타베이스로부터 찾아내는 연산이다[1][2][3]. 이러한 유사 검색은 데이타 마이닝(data

mining) 및 데이타 웨어하우징(data warehousing) 분야에서 중요한 연산으로 사용된다[4].

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][3][4][5][6].

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][3] 등에서는 변환을 지원하지 않았으나, 이후에는 정규화(normalization)[2][5][6][7], 이동 평균(moving average)[4][8], 타임 워핑(time warping)[9][10][11] 등의 다양한 변환을 지원하는 방법들이 제안되었다.

본 논문에서는 시계열 데이타베이스로부터 주어진 질의 시퀀스와 유사한 모양을 갖는 시퀀스를 검색하는 문제를 다루고자 한다. 본 논문에서는 이러한 검색 문제를 모양 기반 검색(shape-based retrieval)이라 정의한다.

참고 문헌 [12]에서는 모양 정의 언어(shape definition language: SDL)를 제안하고, 질의 시 사용자가 이를 이용하여 특정한 패턴을 정의하도록 하였다. 질의 처리에서는 계층적 인덱스 구조(hierarchical index structure)를 사용함으로써 SDL로 정의된 패턴을 만족하는 서브시퀀스들을 효과적으로 검색한다. 이 방식은 시퀀스의 값을 심볼의 형태로 바꾼 후, 원래의 값은 무시한 채 심볼들만을 대상으로 검색을 수행한다. 따라서 거의 유사한 두 값이 서로 다른 심볼들로 변환되는 경우, 이 두 값을 각각 포함하는 유사한 두 시퀀스가 유사하지 않은 것으로 간주되는 문제점을 갖는다.

참고 문헌 [13]에서는 유사-기반 패턴 질의 모델인 랜드마크 모델(landmark model)을 제안하였다. 랜드마크 모델은 두 시퀀스들로부터 중요한 의미를 갖는 요소인 랜드마크들을 각각 추출하고, 실제 시퀀스 요소 값이 아닌 랜드마크들을 대상으로 하여 유사한 정도를 측정한다. 이 모델을 이용함으로써 실제 요소 값에 관계없이 사람들이 생각하는 바와 같은 방식으로 모양이 유사한 시퀀스들을 찾아 낼 수 있다. 그러나 랜드마크들은 각 응용에 따라 다르므로, 해당 응용에 적합한 랜드마크를 찾아내는 것은 간단한 일이 아니다.

본 연구에서는 모양 기반 검색을 해결하기 위하여 기존의 값 기반 검색에서 사용하던 다양한 형태의 변환들의 조합을 이용하고자 한다. 즉, 시프팅 변환, 스케일링 변환, 타임 워핑 변환, 이동 평균 변환 등 다양한 형태의 변환을 동시에 지원하는 유사 검색 모델을 제안함으로써 효과적인 모양 기반 검색을 가능하도록 한다. 기존의 연구에서 이러한 변환을 개별적으로 지원하는 기법들은 다수 제안된 바 있으나, 이들의 다양한 조합을 통하여 위와 같은 모양 기반 검색을 지원하는 기법은 제안된 바 없다.

유사 검색은 전체 검색과 서브시퀀스 검색으로 구분된다[1]. 전체 검색은 시퀀스들과 질의 시퀀스의 길이가 동일하다는 조건 하에 수행되며, 질의 시퀀스와 유사한 시퀀스를 검색한다. 반면, 서브시퀀스 검색은 이러한 조건이 불필요하며, 질의 시퀀스와 유사한 서브시퀀스를 포함하는 시퀀스와 그 시퀀스 내에서의 해당 서브시퀀스의 시작 위치를 검색한다. 모든 시퀀스들의 길이가 동일하다는 전제는 비현실적이므로, 서브시퀀스 검색은 전체 검색에 비하여 다양한 실제 분야에 적용될 수 있다.

본 논문에서는 모양 기반 서브시퀀스 검색을 위한 새로운 기법을 제안한다. 본 논문의 공헌은 다음과 같다. (1) 효과적인 모양 기반 서브시퀀스 검색을 위한 새로운 유사 검색 모델을 정의한다. (2) 이 모델을 기반으로 하는 모양 기반 유사 검색을 효율적으로 처리하기 위한 인덱싱 방안을 제안한다. (3) 제안된 인덱싱 방안을 이용한 질의 처리 방안을 제안한다. (4) 제안된 모양 기반 서브시퀀스 검색 기법의 우수성을 규명하기 위하여 다양한 실험들의 수행을 통하여 나타난 검색 결과의 질과 검색 성능을 제시한다.

본 논문의 구성은 다음과 같다. 제 2장에서는 논의 전개에 필요한 용어 및 기호, 그리고 본 논문에서 제시하는 유사 모델을 정의한다. 제 3장에서는 제안한 유사 모델을 기반으로 하는 질의를 효과적으로 처리하기 위한 인덱싱 기법을 제시한다. 제 4장에서는 제시한 인덱싱 기법을 기반으로 하는 질의 처리 기법을 제안한다. 제 5장에서는 제안하는 기법의 우수성을 규명하기 위한 성능 평가 결과를 제시한다. 끝으로, 제 6장에서는 본 논문을 요약하고, 결론을 내린다.

## 2. 문제 정의

본 장에서는 본 연구에서 해결하고자 하는 문제를 정의한다. 먼저, 제2.1절에서는 본 논문에서 사용하는 용어 및 기호를 정의한다. 제2.2절에서는 본 논문에서 채택하는 유사 모델(similarity model)과 모양 기반 유사 서브시퀀스 검색 문제를 정의한다.

### 2.1 기호 및 용어 정의

표 2.1은 본 논문에서 빈번하게 사용하는 기본적인 기

호를 정리한 것이다. 데이타베이스 내에 저장된 시퀀스를 데이타 시퀀스라 하고, 질의에 주어지는 시퀀스를 질의 시퀀스라 한다.

표 2.1 기호 정의

| 기호 | 정의 |
|---|---|
| S=(s[i]) | 데이타 시퀀스 (0≤i<Len(S)), Len(S)는 S에 포함되는 요소 값의 수 |
| X=(x[i]) | S에 포함되는 임의의 서브시퀀스 (0≤i<Len(X)≤Len(S)) |
| Q=(q[i]) | 질의 시퀀스 (0≤i<Len(Q)) |
| ε | 유사 허용치 |
| First(S) | 시퀀스 S의 첫 번째 요소 값 s[0] |
| Rest(S) | 시퀀스 S에서 s[0]을 제외한 요소들로 구성된 시퀀스 (s[1], s[2], ..., s[Len(S)-1]) |
| Max(S) | 시퀀스 S내의 크기가 최대인 요소 값 |
| Min(S) | 시퀀스 S에서 크기가 최소인 요소 값 |
| () | 요소가 존재하지 않는 널 시퀀스(null sequence) |
| max(a, b) | a와 b의 값 중 최대 값 |
| min(a, b, c) | a, b, c의 값 중 최소 값 |

정의 1:

시퀀스 S=(s[i])(0≤i<Len(S))를 정규화(normalization) 변환한 시퀀스 Norm(S)=(s'[i]) (0≤i<Len(S))는 다음과 같이 정의된다[2][1].

$$s'[i] = \frac{s[i] - \frac{Max(S)+Min(S)}{2}}{\frac{Max(S)-Min(S)}{2}}$$

□

정규화 변환을 통하여 해당 시퀀스가 갖는 요소 값의 절대적인 크기를 무시할 수 있다. 따라서 정규화 변환은 요소 값의 크기는 서로 다르지만 변화하는 패턴이 유사한 시퀀스들을 파악하는데 매우 유용하다. 예를 들어, 그림 2.1에 나타난 두 개의 시퀀스 S1과 S2는 각각 서로 다른 요소 값들을 가지지만, 정규화 변환을 통하여 동일한 시퀀스 Norm(S1)과 Norm(S2)로 변환된다.

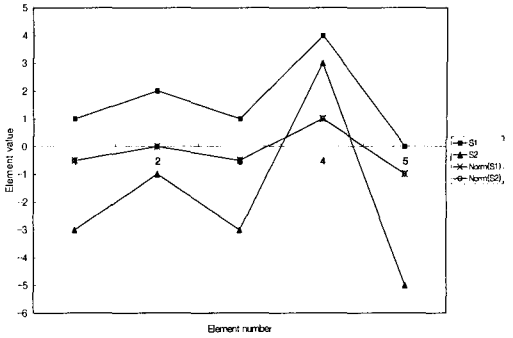


그림 2.1 정규화 변환의 예

정의 2:

시퀀스 S=(s[i])(0≤i<Len(S))를 이동 평균 계수(moving average coefficient) k (1≤k<Len(S))로 이동 평균(moving average) 변환한 시퀀스 $MV_k(S)=(s_k[j])$ (0≤j<Len(S)-k+1)는 다음과 같이 정의된다[14][15].

$$s_k[j] = \frac{1}{k} \times (s[j]+s[j+1]+\cdots\cdots+s[j+k-1]) = \frac{1}{k} \times \sum_{i=j}^{j+k-1} s[i]$$

□

정의 2에 나타난 바와 같이, 이동 평균 변환은 시퀀스의 연속되는 k개의 요소 값들의 평균값들을 순차적으로 나열하는 변환이다. 이동 평균 변환을 통하여 시퀀스 내에서 나타나는 잡음(noise)의 영향을 제거할 수 있다. 따라서 이동 평균 변환은 잡음의 영향 없이 전체적인 변화 경향이 유사한 시퀀스들을 파악하는데 매우 유용하다. 이동 평균 계수 k는 해당 응용에서 잡음의 영향을 줄이고자 하는 정도에 따라 적절하게 선택된다. 예를 들어, 그림 2.2는 시퀀스 S와 이를 4-, 8-이동 평균 변환한 시퀀스 $MV_4(S)$, $MV_8(S)$를 나타낸 것이다. 이동 평균 계수가 커짐에 따라 시퀀스 내의 잡음의 영향이 감소함을 볼 수 있다.

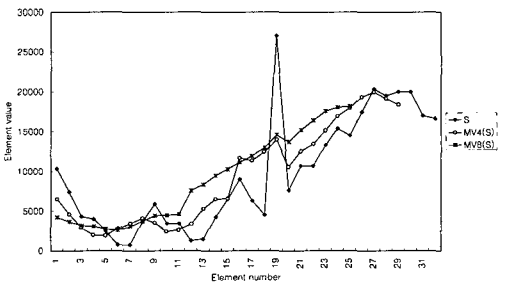


그림 2.2 이동 평균 변환의 예

1) 정규화 변환의 정의로서 S내의 요소 값들의 평균 avg(S)와 표준 편차 std(S)를 이용하는 s'[i]=(s[i]-avg(S))/std(S)를 이용할 수도 있다[4][6]. 그러나 본 연구에서는 정규화 변환 후의 요소 값의 상한과 하한을 각각 +1과 -1로 고정하기 위하여 정의 1을 사용한다. 변환 후의 모든 요소 값들이 +1과 -1 사이에만 분포하게 되므로, 제3장에서 언급하는 도메인 분류(categorization)를 통한 서브시퀀스 트리의 압축 과정에서 좋은 효과를 얻을 수 있다.

정의 3:

길이 n을 갖는 두 시퀀스 S와 Q의 유사한 정도를 측정하기 위한 거리 함수 $L_p$는 다음과 같이 정의된다. 여기서, $L_1$은 맨하탄 거리(Manhattan distance), $L_2$는 유클리드 거리(Euclidean distance), $L_\infty$은 대응되는 각 쌍의 거리 중 최대 거리를 의미한다[16].

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

□

거리 함수 $L_p$는 현재 많은 응용에서 널리 사용되고 있으나, 비교 대상이 되는 두 시퀀스의 길이가 같아야 한다는 제한이 있다[11][17]. 타임 워핑은 시퀀스내의 각 요소 값을 임의의 수만큼 반복시킴으로써 서로 다른 길이의 두 시퀀스간의 유사한 정도를 측정할 수 있도록 허용하는 변환이다[10]. 예를 들어, 서로 다른 두 시퀀스 S1=⟨20, 21, 21, 20, 20, 23, 23, 23⟩과 S2=⟨20, 20, 21, 20, 23⟩가 타임 워핑 변환에 의하여 동일한 시퀀스 S3=⟨20, 20, 21, 21, 20, 20, 23, 23, 23⟩으로 변환된다.

정의 4:

타임 워핑 변환 후의 두 시퀀스 S와 Q간의 거리인 타임 워핑 거리(time warping distance) $D_{tw}$는 다음과 같이 재귀적으로 정의된다[18]. 여기서, $D_{base}$는 기본 거리 함수로서 $L_p$ 중 응용에 적합한 임의의 것을 선택하여 사용할 수 있다.

(1) $D_{tw}(\langle\rangle, \langle\rangle)=0$,

(2) $D_{tw}(S, \langle\rangle)=D_{tw}(\langle\rangle, Q)=\infty$,

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

□

타임 워핑은 데이타베이스내의 시퀀스들의 길이가 서로 달라서 유클리드 거리를 이용하여 유사 정도를 직접 측정할 수 없는 경우에 매우 유용하다. 타임 워핑 변환은 정의 4(3)에서와 같이 두 시퀀스의 거리 차를 최소화하기 위하여 한 시퀀스 내의 임의의 요소를 반복시킴으로써 이 요소가 다른 시퀀스의 다수의 요소들과 대응되는 것을 허용한다.

### 2.2 유사 모델

본 연구의 주요 목적은 질의에서 주어진 시퀀스와 유사한 모양을 갖는 서브시퀀스를 효과적으로 검색하는 기능을 지원하는 것이다. 본 연구에서는 이러한 모양 기반 유사 검색을 지원하기 위하여, 앞에서 언급한 정규화 변환, 이동 평균 변환, 타임 워핑 변환을 모두 지원하는 다음과 같은 유사 정도 측정 함수를 채택한다.

정의 5:

두 시퀀스 혹은 서브시퀀스 S와 Q간의 유사한 정도 D(S, Q)는 다음과 같이 정의된다.

$D(S, Q)=D_{tw}(Norm(MV_k(S)), Norm(MV_k(Q)))$

□

정의 5에 나타난 바와 같이, S와 Q간의 유사한 정도 D(S, Q)는 S와 Q를 각각 (1) k-이동 평균 변환, (2) 정규화 변환, (3) 타임 워핑 변환을 차례로 거친 후의 거리로 정의된다. 특히, 타임 워핑 거리 계산에서는 거리 함수 $D_{base}$로서 $L_\infty$를 사용한다. 이를 위하여 정의 4에 나타난 타임 워핑 거리는 다음 정의 6과 같은 형태로 변형된다.

정의 6:

(1) $D_{tw}(\langle\rangle, \langle\rangle)=0$,

(2) $D_{tw}(S, \langle\rangle)=D_{tw}(\langle\rangle, Q)=\infty$,

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

□

$D_{tw}$의 값은 동적 프로그래밍 기법(dynamic programming)에 의한 거리 축적 테이블(cumulative distance table)을 이용하여 효율적으로 계산될 수 있다[9][18]. 다음의 그림 2.3은 거리 축적 테이블을 이용한 두 시퀀스 S와 Q의 타임 워핑 거리 계산 예를 보인다.

| | | | |
|---|---|---|---|
| 6 | 4 | 3 | ***3*** |
| 6 | 4 | ***3*** | 3 |
| 7 | 4 | ***3*** | 4 |
| 6 | 3 | ***2*** | 3 |
| 5 | 2 | ***1*** | 2 |
| 4 | ***1*** | 1 | 1 |
| S \ Q | 3 | 4 | 3 |

그림 2.3 S=⟨4, 5, 6, 7, 6, 6⟩와 Q=⟨3, 4, 3⟩의 타임 워핑 거리 계산 예

시계열 데이타베이스를 위한 모양 기반 유사 검색 문제를 다음과 같이 정의한다. 질의 시퀀스 Q, 유사 허용치 ε, 이동 평균 계수 k가 주어지면, D(X, Q)(= $D_{tw}(Norm(MV_k(X)), Norm(MV_k(Q)))$)의 값이 ε이하인 데이타베이스 내의 서브시퀀스 X를 찾고, X를 포함하는 데이타 시퀀스 S와 S내에서의 서브시퀀스 X의 시작 위치를 반환한다. D(X, Q)의 값이 ε이하라는 의미는 $Norm(MV_k(X))$를 타임 워핑 변환한 시퀀스의 각 요소가 $Norm(MV_k(Q))$를 타임 워핑 변환한 시퀀스의 대응

되는 요소의 일정 범위 ε내에' 존재함을 의미한다.

본 논문에서 제안하는 유사 모델을 기반으로 한 모양 기반 유사 시퀀스 검색 기법은 시프팅(shifting), 스케일링(scaling), 타임 워핑(time warping) 등 다양한 변환을 지원할 뿐만 아니라, 이동 평균 변환을 지원하므로 시퀀스 내에서 발생하는 잡음의 영향도 최소화 할 수 있다.

$D_{base}$로서 $L_1$을 사용하는 참고 문헌 [9][10][11]과 달리, 본 연구에서 $L_\infty$를 사용하는 주된 이유는 사용자의 질의 작성의 부담을 덜도록 하기 위해서이다[17]. $L_1$을 사용하는 경우, 타임 워핑 거리는 변환된 두 시퀀스의 각 대응되는 요소 쌍의 거리들의 합으로 나타나므로 질의 시퀀스와 데이타 시퀀스의 길이에 큰 영향을 받는다. 본 논문에서 고려하는 서브시퀀스 매칭 문제에서는 같은 데이타 시퀀스로부터 추출되는 서브시퀀스들이라 할 지라도 그 길이 차가 매우 크므로 이러한 문제는 더욱 심각하게 나타난다[2]. 따라서 질의를 작성하는 사용자가 해당 데이타베이스 특성에 맞는 적절한 ε을 결정한다는 것은 매우 어려운 일이다. 특히, 동적인 환경에서는 시퀀스의 길이가 계속 변경되므로 올바른 ε를 결정하는 것이 사실상 불가능하다. 반면, $L_\infty$를 사용하는 경우, 시퀀스의 길이에 영향을 받지 않고 일관된 ε을 사용할 수 있으므로 이러한 질의 작성의 부담을 덜 수 있다.

## 3. 인덱싱 방안

본 장에서는 제 2장에서 제안한 유사 모델을 기반으로 하는 모양 기반 검색 질의를 효과적으로 처리하기 위한 인덱싱 방안에 관하여 논의한다. 제 3.1절에서는 인덱스 구조로서 사용하는 접미어 트리(suffix tree)에 대하여 설명하고, 제 3.2절에서는 접미어 트리를 이용한 인덱싱 전략에 관하여 논의한다. 제 3.3절에서는 구체적인 인덱스 구성 절차를 기술한다.

### 3.1 접미어 트리

접미어 트라이(suffix trie)[19]는 각 시퀀스에 속하는 모든 접미어(suffix)들을 대상으로 트라이(trie)를 구성한 것이다. 접미어 트리(suffix tree)는 접미어 트라이에서 차수(degree)가 1인 연속된 노드들을 하나의 노드로 결합한 구조이다. 접미어 트리는 다수의 시퀀스들을 인덱싱하기 위하여 사용되며, 주어진 질의 시퀀스와 정확하게 매치되는 서브시퀀스의 위치를 신속하게 찾도록 하는데 유용하다[19]. 참고 문헌 [11]에서는 접미어 트리를 이용하여 타임 워핑 변환을 지원하는 효과적인 서브시퀀스 검색 방법을 제안한 바 있다.

한 시퀀스에 속하는 각 접미어는 트리 내의 리프 노드(leaf node)와 대응된다. 예를 들어, 시퀀스 Si=(s[0], s[1], ...., s[Len(Si)-1])의 한 접미어 (s[j], s[j+1], ....., s[Len(Si)-1])는 라벨 (Si, j)를 갖는 리프 노드로서 표현된다. 접미어 트리의 각 에지(edge)들도 라벨을 가지며, 트리의 루트로부터 라벨 (Si, j)를 가지는 리프 노드까지 경로상의 에지 라벨들을 모두 접합(concatenation)하면 접미어 (s[j], s[j+1], ....., s[Len(Si)-1])가 된다. 루트로부터 한 내부 노드(internal node)까지 경로상의 에지 라벨들을 모두 접합한 결과는 그 내부 노드 아래에 존재하는 모든 리프 노드와 대응되는 접미어들의 최장 공통 접두어(longest common prefix)가 된다. 그림 3.1은 두 개의 시퀀스 S1=(1.1, 2.5, 2.5, 1.1)와 S2=(1.1, 2.5)를 대상으로 구성된 접미어 트리를 예로 나타낸 것이다.

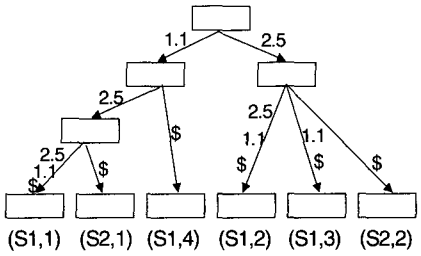


그림 3.1 접미어 트리의 예

접미어 트리는 저장하는 접미어들이 많은 공통 접두어 서브시퀀스를 가질 때, 좋은 압축 효과를 갖는다. 그러나 시퀀스의 요소 값은 실수의 타입을 가지므로 접미어들이 공통의 접두어 서브시퀀스를 가질 가능성은 매우 낮다. 참고 문헌 [11]에서는 이러한 문제를 해결하기 위하여 도메인 분류(categorization)를 사용하는 방법을 제안한 바 있다. 도메인 분류는 요소 값을 심볼로 변환하기 위하여 요소 값의 도메인을 여러 범위들로 분할하는 작업이다. 서로 다른 요소 값이라도 같은 범위에 속하게 되면, 동일한 심볼로 변환된다. 이러한 심볼로 변환된 시퀀스들을 대상으로 접미어 트리를 구성하는 경우, 접미어들이 공통의 접두어 서브시퀀스를 가질 가능성이 상대적으로 높아진다.

2) 예를 들어, 길이가 1000인 데이타 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 $L_2$를 $D_{base}$로 사용하는 경우, 유사성을 판별하기 위하여 $D_{tw}(q, s1)$과 $D_{tw}(q, s2)$를 같은 ε값을 기준으로 비교한다는 것은 매우 불합리하다.

### 3.2 인덱스 구성 전략

접미어 트리 내에 저장된 시퀀스 S의 한 접미어 Sa=(s[i], s[i+1], ...., s[Len(S)-1])과 Sa의 임의의 접두어 서브시퀀스 Sb=(s[i], s[i+1], ...., s[j]) (j<Len(Sa))를 고려해 보자. Sb는 Sa의 접두어이므로 Sb의 모든 요소 값들은 Sa내에 포함된다. 따라서 Sa만을 참조해서도 질의 시퀀스 Q와 타임 워핑 거리가 유사 허용치 이하인 서브시퀀스 Sb를 검색할 수 있다. 그러므로 유사 서브시퀀스 검색에서 변환을 지원하지 않는 경우에는 모든 시퀀스들의 가능한 접미어들만을 접미어 트리 내에 저장함으로써 임의의 유사한 서브시퀀스를 검색할 수 있다.

본 논문에서 채택하는 유사 모델에서는 정규화 변환을 지원한다. 정의 1에 나타난 바와 같이, 한 서브시퀀스 S를 정규화 변환하기 위해서는 Max(S)와 Min(S)를 사용한다. 위의 예에서 사용된 Sa와 Sb를 다시 고려하면, Max(Sa)와 Max(Sb), 그리고 Min(Sa)와 Min(Sb)는 일반적으로 서로 다른 값을 갖는다. 이 결과, Norm(Sb)는 Norm(Sa)의 접두어가 아니므로 Norm(Sa)만을 참조하여 Norm(Q)와의 타임 워핑 거리가 유사 허용치 이하인 Norm(Sb)를 찾을 수 없다. 따라서 정규화 변환을 지원하는 경우에는 접미어 뿐만 아니라 발생 가능한 모든 서브시퀀스들을 개별적으로 인덱스 내에 저장해야 한다. 본 논문에서는 접미어 트리와 구조적으로 동일하며, 각 시퀀스 내의 모든 가능한 서브시퀀스들을 포함하는 인덱스를 서브시퀀스 트리(subsequence tree)라 정의한다.

길이가 n인 시퀀스에 대하여 접미어 트리는 n개의 접미어들만을 인덱싱의 대상으로 하는 반면, 서브시퀀스 트리는 n*(n+1)/2 개의 서브시퀀스들을 인덱싱의 대상으로 한다. 그러나 다음과 같은 특성으로 인하여 서브시퀀스 트리가 같은 시퀀스들을 대상으로 구성된 접미어 트리에 비하여 지나치게 커지는 현상은 발생하지 않는다. 첫째, 정의 1에 의한 정규화를 통하여 변환 후의 모든 요소 값들은 −1에서 +1사이에만 존재하게 된다. 둘째, 같은 시퀀스에 속하는 접미어들은 공통되는 접두어를 가질 가능성이 낮은 반면, 같은 시퀀스에 속하는 서브시퀀스들은 많은 공통되는 접두어를 갖는다. 따라서 원래의 요소 값이 다른 경우에도 정규화 변환 후에는 도메인 분류에 의하여 동일한 심볼로 변환될 가능성이 높아지므로 서브시퀀스들은 많은 공통 접두어를 갖게 된다. 실험 결과에 의하면, 같은 데이타베이스를 대상으로 구성된 접미어 트리와 서브시퀀스 트리의 내부 노드 수는 거의 동일하며, 단지 리프 노드 수에 대해서만 차이가 있는 것으로 나타났다.

### 3.3 서브시퀀스 트리 구성 절차

시계열 데이타베이스를 대상으로 서브시퀀스 트리를 구성하는 절차는 (1) 이동 평균 변환, (2) 서브시퀀스 추출, (3) 정규화 변환, (4) 심볼 변환, (5) 트리 구성의 다섯 단계로 이루어진다.

단계 1: 이동 평균 변환

시계열 데이타베이스 내의 각 시퀀스 S에 대하여 k-이동 평균 변환을 수행한다. 여기서, 이동 평균 계수 k 값은 해당 응용에 적합한 것을 선정한다. S를 k-이동 평균한 시퀀스를 $MV_k(S)$라 하자.

단계 2: 서브시퀀스 추출

각 $MV_k(S)$로부터 모든 가능한 서브시퀀스 X를 추출한다[3]. 이때, 추출될 서브시퀀스의 최소 길이 L을 지정할 수 있다. 즉, 너무 짧은 X가 질의 결과로서 큰 의미가 없는 경우에는 L 이상의 길이를 갖는 서브시퀀스만을 추출하도록 한다. 여기서, L 값은 해당 응용에 적합한 것을 선정한다.

단계 3: 정규화 변환

각 서브시퀀스 X에 대하여 고유의 Min(X)와 Max(X) 값을 이용함으로써 정규화 변환을 수행한다. X를 정규화 변환한 것을 Norm(X)라 하자.

단계 4: 도메인 분류를 이용한 심볼 변환

각 Norm(X)에 대하여 도메인 분류를 이용함으로써 심볼 변환을 수행한다. 즉, Norm(X)의 각 요소 값이 도메인 분류에 의하여 정의된 범위들 중 어느 곳에 속하는가 하는 것을 파악한 후, 이 범위와 대응되는 심볼로 변환시킨다. 여기서, 도메인 분류에 의하여 정의된 범위들의 수 C는 해당 응용에 적합한 것을 선정한다.

단계 5: 트리 구성

심볼로 변환된 서브시퀀스들을 대상으로 서브시퀀스 트리를 구성한다. 대용량의 데이타베이스 환경에서는 서브시퀀스 트리가 디스크에 존재하게 되므로, 서브시퀀스 트리의 효과적인 구성을 위하여 디스크 기반 접미어 트리 구성 알고리즘[20]을 활용한다.

## 4. 질의 처리

본 장에서는 제 3장에서 제안한 인덱싱 전략을 기반으로 하는 모양 기반 유사 서브시퀀스 검색 질의를 효과적으로 처리하는 알고리즘을 제시하고, 그 검색 성능을 분석한다.

### 4.1 질의 처리 알고리즘

먼저, 질의에서는 다음의 두 가지 방식에 의한 이동

3) 즉, X는 S의 서브시퀀스가 아닌 MVk(S)의 서브시퀀스이다.

평균된 시퀀스를 입력받는 것을 전제로 한다. 첫 번째는 사용자가 질의 패턴을 직접 작성하여 질의 시퀀스로 사용하는 방식이다. 즉, 사용자가 마우스 혹은 키보드를 이용하여 질의 시퀀스에 속하는 요소 값을 결정한다. 이 경우, 사용자가 원하는 시퀀스의 모양을 직접 그리는 상황이므로, 실질적으로 잡음이 발생하지 않는다. 두 번째는 사용자가 데이타베이스 내에서 선택한 이동 평균된 시퀀스로부터 일부의 서브시퀀스를 추출하여 질의 시퀀스로 사용하는 방식이다. 두 가지 경우에서 모두 실질적으로는 이미 이동 평균된(잡음이 존재하지 않는) 질의 시퀀스를 대상으로 하므로, 이를 다시 정규화 변환하여 아래의 질의 처리 알고리즘을 적용한다.

그림 4.1은 서브시퀀스 트리를 이용하여 질의 시퀀스 Q와 $D_{tw}$가 ε이내인 유사한 서브시퀀스들을 데이타베이스로부터 검색하는 알고리즘 S-ST를 나타낸 것이다. 여기서, 질의 시퀀스와 서브시퀀스 트리 내의 모든 시퀀스들은 이미 이동 평균 변환과 정규화 변환을 모두 끝낸 것임에 유의해야 한다.

제 3.3절에서 설명한 바와 같이 서브시퀀스 트리는 심볼로 변환된 서브시퀀스들을 대상으로 구성되어 있으므로, 서브시퀀스와 질의 사이의 타임 워핑 거리 $D_{tw}$를 직접 계산할 수 없다. 따라서 S-ST에서는 심볼로 변환된 서브시퀀스와 질의 Q 사이의 타임 워핑 거리 계산을 위하여, $D_{tw}$의 하한 함수인 $D_{tw\text{-}lb}$를 다음과 같이 재귀적으로 정의한다[11].

정의 7:

(1) $D_{tw\text{-}lb}((), ())=0$

(2) $D_{tw\text{-}lb}(CS, ())=D_{tw\text{-}lb}((), Q)=\infty$

(3) $D_{tw\text{-}lb}(CS, Q)=\max(D_{base\text{-}lb}(First(CS), First(Q)),$
$\min(D_{tw\text{-}lb}(CS, Rest(Q)), D_{tw\text{-}lb}(Rest(CS), Q),$
$D_{tw\text{-}lb}(Rest(CS), Rest(Q))))$

(4) $D_{base\text{-}lb}(A, b)=0$ (if $A.lb \le b \le A.ub$)
$=b-A.ub$ (if $b > A.ub$)
$=A.lb-b$ (if $b < A.lb$)

□

여기서, CS는 심볼로 변환된 서브시퀀스를 의미한다. 또한, A는 First(CS)에 해당하는 심볼을 나타내며, b는 First(Q)에 해당하는 실제의 실수 값을 나타낸다. A.lb와 A.ub는 각각 A가 속한 도메인의 최소 요소 값과 최대 요소 값을 나타낸다.

이와 같이, 제안된 기법은 인덱싱의 대상인 서브시퀀스들을 심볼로 변환하지만, 질의 시퀀스의 원래 요소 값을 기반으로 검색을 수행한다. 따라서 참고 문헌 [12]에서와는 달리 유사한 두 값이 서로 다른 심볼로 변환되는 경우에도 유사한 두 시퀀스가 유사하지 않은 것으로 간주되는 현상은 발생되지 않는다.

S-ST의 실질적인 탐색 과정은 VisitNode-and-FindAnswers의 수행을 통하여 실행된다. VisitNode-and-FindAnswers는 하나의 노드를 방문하면, 최종 결과를 얻기 위하여 그 노드의 모든 자식 노드를 방문함으로써 각 노드에서 그 이상의 깊이 탐색이 필요한가의 여부를 판별한다. 예를 들어, 알고리즘이 노드 N을 방문하고 그 자식 노드 $CN_i$를 탐색하는 과정은 다음과 같다.

첫 번째 과정에서는 질의 Q와 label(N, $CN_i$)을 위한 거리 축적 테이블 $CT_i$를 생성한다. 만일, 노드 N이 루트 노드라면 테이블은 처음부터 생성되며, 루트 노드가 아니라면, 노드 N에 대하여 생성된 테이블 T 위에 새로운 행(row)이 쌓이게 된다. 함수 Addrow()는 $D_{tw\text{-}lb}$를 사용하여 거리 축적 테이블을 생성한다. S-ST에서는 테이블에 새로운 행이 쌓일 때마다 다음 과정에 의하여 더 이상의 깊이 우선 트리 탐색이 필요한지를 판별한다. 즉, 새로이 쌓인 행의 각 요소 중 최소 값 mDist를 구하고, 그 mDist가 ε보다 크면 더 이상의 깊이 탐색이 의미 없다고 판별한다. 따라서 mDist가 ε보다 크면, 깊이 우선 트리 탐색이 중지되고, 그 다음 탐색은 노드 N의 다음 자식 노드인 $CN_{i+1}$로 이어진다. 두 번째 과정에서는 노드 $CN_i$에서 새로운 결과를 얻기 위한 가능성을 판별한다. $CT_i$의 새로이 생성된 행의 마지막 요소 값을 검사하여, 그 값이 ε보다 작거나 같고 방문 중인 노드가 리프 노드이면, 리프 노드가 표현하는 서브시퀀스를 answerSet에 넣는다. 그 후, 다음 탐색은 노드 N의 다음 자식 노드인 $CN_{i+1}$로 이어진다. 세 번째 과정에서는 새로운 결과를 얻기 위하여 노드 $CN_i$의 다음 자식 노드로부터 깊이 우선 트리 탐색을 재귀적으로 진행한다.

이와 같이 VisitNode-and-FindAnswers에 의하여 반환된 모든 결과는 하한 함수인 $D_{tw\text{-}lb}$를 이용하여 얻어진 것이므로, 실제의 타임 워핑 거리가 ε보다 큰 서브시퀀스가 결과에 포함될 수 있다. 이와 같은 서브시퀀스들을 착오 채택(false alarms)이라 부른다[10].

S-ST에서는 착오 채택된 서브시퀀스를 제거하기 위하여 다음과 같은 후처리(post processing)를 수행한다. 함수 *Postprocess*()에서는 VisitNode-and-FindAnswers에서 반환된 모든 결과에 대하여 실제의 서브시퀀스 S를 액세스하고, 정의 5에 나타난 유사도 함수 D(S, Q)를 계산한다. D(S, Q)가 ε보다 작거나 같은 서브시퀀스만이 actual-ans에 포함되어 최종 결과로 반환된다.

```
Algorithm S-ST
candidate-ans ← VisitNode-and-FindAnswers(root R, Q, ε,
                emptyTable);
actual-ans ← PostProcess(candidate-ans);
return actual-ans;
End

Algorithm VisitNode-and-FindAnswers(Node N, Q, ε,
cumDistTable T)
For i ← 1 to |CN| do {
    For j ← 1 to |label(N, CNi)| do {
        CTi ← Addrow(T, Q, label(N, CNi)[j], Dtw-lb());
        Let mDist be the minimum column value of the new row;
        If mdist > ε goto next-CN;
    }
    Let dist be the last column value of the new row;
    If (CNi is a leaf node && dist ≤ ε) then {
        answerSet ← FindAnswer(CNi);
        goto next-CN;
    }
    answerSet ← answerSet ∪
    VisitNode-and-FindAnswers(CNi, Q, ε, CTi)
    label: next-CN ;
}
return answerSet;
End
```

그림 4.1 S-ST: 서브시퀀스 트리를 이용한 유사검색 알고리즘

### 4.2 알고리즘 분석

S-ST의 질의 처리 성능을 보이기 전에 우선 기본적인 순차 검색(sequential scanning)에 의한 질의 처리 과정을 간단히 설명한다. 순차 검색에서는 데이타베이스 내의 각 시퀀스를 읽어들여 k-이동 평균한 후, 각 이동 평균된 시퀀스에 포함된 모든 서브시퀀스를 추출하여 정규화 변환한다. 다음, 정규화 변환된 질의 시퀀스와 서브시퀀스의 $D_{tw}$를 계산하기 위하여 거리 축적 테이블을 생성한다. 질의 시퀀스 Q와 길이 L의 서브시퀀스를 위한 거리 축적 테이블의 생성 비용은 $O(L|Q|)$이다[20]. M개의 시퀀스를 대상으로 하는 경우, M개 시퀀스의 평균 길이를 $L'$이라 하면, $ML'(L'+1)/2$개의 서브시퀀스가 생성되고, 각 서브시퀀스의 평균 길이는 $(L'+2)/3$이다. 따라서 순차 검색에 의한 질의 처리 비용은 $O(ML'^3|Q|)$이다.

S-ST는 순차 검색에 비하여 질의를 처리하는 비용을 크게 줄일 수 있다. 그 이유는 공통 접두어를 갖는 다수의 서브시퀀스들이 거리 축적 테이블을 공유함으로써 얻어지는 효과와 탐색 중의 가지치기(branch-pruning) 현상에 의한 것이다. S-ST의 비용은 $O(ML'^3|Q|/R_dR_p + nL'|Q|)$이다. 좌측 수식은 트리 탐색을 위한 것이며, 우측 수식은 후처리를 위한 것이다. 여기서, $R_d(\geq 1)$는 거리 축적 테이블의 공유에서 얻어지는 감소 계수(reduction factor)를 나타내며, $R_p(\geq 1)$는 가지치기 현상에서 얻어지는 감소 계수를 나타낸다. 또한, n은 후처리를 필요로 하는 서브시퀀스 개수를 나타낸다.

$R_d$의 값은 서브시퀀스 사이에 발생하는 공통 접두어의 길이와 개수가 증가할 수록 크기가 커진다. 즉, $R_d$는 서브시퀀스의 요소 값 분포에 의존하는 값으로서, 서브시퀀스의 요소 값으로 도메인 분류에 의한 심볼을 사용하는 S-ST에서는 상당한 크기의 $R_d$를 기대할 수 있다. 한편, $R_p$는 사용자가 요구하는 해의 개수에 의존하는 값으로서, 유사 허용치 ε이 작아지면, 상대적으로 $R_p$는 증가한다. 만일, ε이 매우 작아, 사용자가 하나나 둘 정도의 유사 서브시퀀스의 검색만을 원한다면, 인덱스의 아주 적은 부분만을 검색하게 되어 질의 처리 시간이 현저히 감소한다. 극단의 경우, ε이 매우 커서 모든 서브시퀀스가 검색 대상이 되는 경우에는 인덱스의 모든 노드가 방문되어 $R_p=1$이 된다. 최악의 경우, 공통 접두어를 갖는 서브시퀀스가 존재하지 않으며, 가지치기가 발생하지 않는 경우에는 $R_d=R_p=1$이 되어 S-ST의 처리 비용은 순차 검색의 경우와 같게 된다.

## 5. 성능 평가

본 장에서는 실제 데이타인 S&P_Data를 이용한 실험을 통하여 서브시퀀스 트리를 이용한 모양 기반 유사 서브시퀀스 검색 알고리즘의 성능을 측정한다. S&P_Data는 미국의 S&P 500 주식 데이타로서, 각 주식의 매일 종가를 기록한 시퀀스들의 집합이다. 제 4장에서 언급한 바와 같이, 질의 시퀀스로는 이동 평균 변환을 취한 주식 데이타 시퀀스로부터 임의로 추출한 서브시퀀스를 사용한다.

그림 5.1은 본 논문에서 제안된 기법을 통하여 수행한 유사 서브시퀀스 검색의 예를 보인다. 질의로서 처음 그림의 이동 평균선 상에 굵은 선으로 표현된 이중 바닥(double bottoms) 패턴의 서브시퀀스를 이용하였으며, 그 후 다섯 개의 그림의 이동 평균선 상에 굵은 선으로 표현된 부분이 검색된 서브시퀀스 결과를 나타낸다. 사용자가 원하는 이중 바닥을 갖는 서브시퀀스들을 실제 요소 값에 관계없이 효과적으로 검색하였음을 볼 수 있다.

S-ST의 검색 성능을 평가하기 위하여 순차 검색 방식을 그 비교 대상으로 하며, 평균 질의 처리 시간과 인덱스 크기를 성능 평가 지수로 사용한다. 이를 위하여 다음과 같은 다섯 종류의 실험을 실시하였다. 시퀀스 데이타는 기본적으로 10-이동 평균 변환된 길이 100의 시

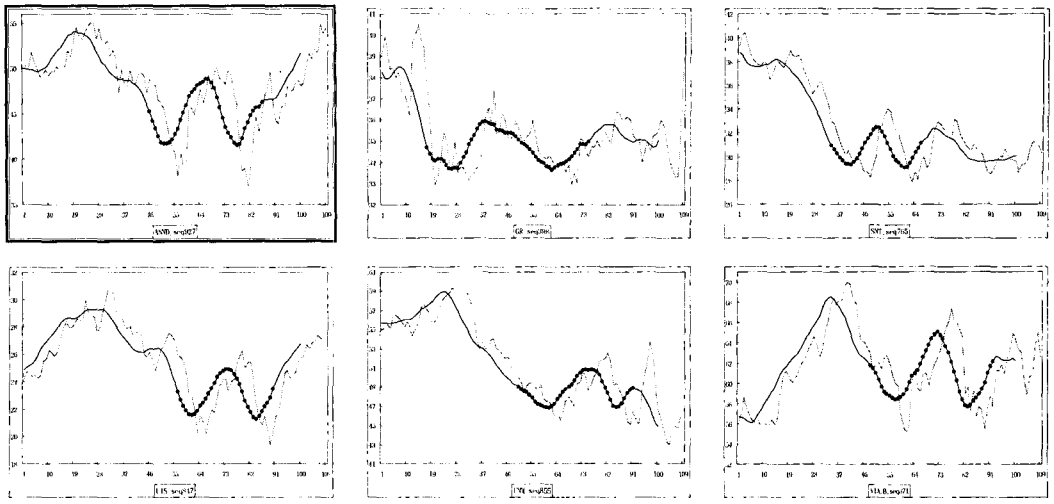

그림 5.1 유사 서브시퀀스의 검색 예

퀀스를 이용하였으며, 개수는 200개의 것을 이용하고, 질의의 평균 길이는 20으로 하였다. 또한, 질의 평균 길이의 절반 이하인 서브시퀀스들은 인덱싱의 대상에서 제외하였다.

실험 1: 표 5.1은 도메인 분류의 개수 C의 변화에 따른 서브시퀀스 트리의 크기 변화와 이에 따른 평균 질의 처리 시간의 변화를 보인다. ε의 값으로는 평균 100개의 최종 결과를 얻기 위한 값(ε=0.113483)을 사용하였다. C의 값이 증가하면 인덱스 크기는 증가하지만, 평균 질의 처리 시간은 감소한다. 그러나 C의 값이 어느 정도의 한계 값을 넘으면, 인덱스 크기와 질의 처리 시간의 변화율이 매우 적어진다. 이 한계 값을 최적의 C 값으로 사용하였다. 예를 들어, 이 경우에는 60을 최적의 도메인 분류 수로 볼 수 있다. 이후의 모든 실험에서는 이 결과를 기반으로 도메인 분류 수를 60으로 하여 질의 처리 시간을 측정한다.

표 5.1. 도메인 분류 수의 변화에 따른 인덱스 크기와 질의 처리 시간의 비교

| # of Categories | Index Size (Kbytes) | Avg. Query Processing Time (sec) |
|---|---|---|
| 10 | 46,131 | 40.6 |
| 20 | 54,651 | 35.8 |
| 40 | 59,751 | 28.0 |
| 60 | 61,746 | 22.6 |
| 80 | 62,802 | 23.8 |
| 100 | 63,457 | 21.2 |

실험 2: 그림 5.2는 유사 허용치 ε의 변화에 따른 질의 처리 시간의 비교 결과를 나타낸다. ε는 평균 10개의 최종 결과를 얻기 위한 값부터 30개, 100개, 300개의 최종 결과를 얻기 위한 값을 사용하였다. 본 실험의 결과, S-ST는 순차 검색 방식에 비하여 유사 허용치의 변화에 따라 약 20배에서 35배까지 높은 성능 향상을 보이고 있다.

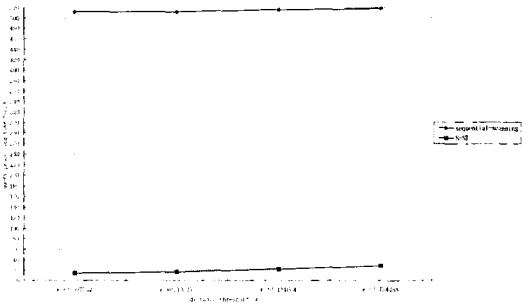

그림 5.2 유사 허용치 변화에 따른 질의 처리 시간의 비교

실험 3: 그림 5.3은 질의 평균 길이의 변화에 따른 질의 처리 시간의 비교 결과를 나타낸다. ε는 각각의 질의 길이에 대하여 평균 100개의 최종 결과를 얻기 위한 값을 사용하였다. 질의의 평균 길이가 증가하여도 S-ST는 순차 검색 방식에 비하여 실험 2와 거의 동일한 성능 향상 결과를 보이고 있다.

다음의 실험 4와 5에서는 주식 시퀀스 데이타의 개수와 평균 길이를 증가시키는 경우에 대한 알고리즘의 성능

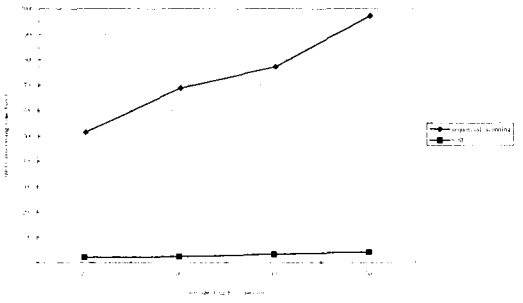

그림 5.3 질의 평균길이 변화에 따른 질의 처리 시간의 비교

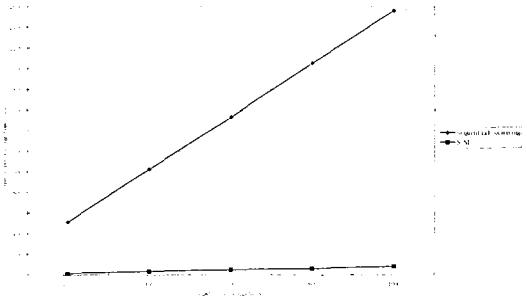

그림 5.4 시퀀스 수 변화에 따른 질의 처리 시간의 비교

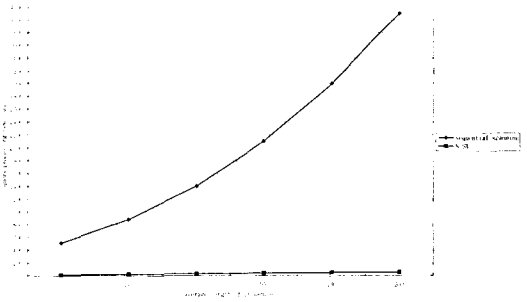

그림 5.5 시퀀스 평균 길이 변화에 따른 질의 처리 시간의 비교

측정 결과를 보인다. 우선, 시퀀스의 길이는 100으로 고정시키고, 시퀀스 개수를 200에서 1000까지 증가시킨다. 다음, 시퀀스의 개수는 200으로 고정시키고, 시퀀스의 평균 길이를 100에서 200까지 증가시킨다. 두 경우 모두 도메인 분류 수는 60으로 하고 평균 질의 길이를 20으로 한다.

실험 4: 그림 5.4는 시퀀스 개수의 변화에 따른 질의 처리 시간의 비교 결과를 나타낸다. ε는 시퀀스 개수 800의 데이타에서 평균 약 100개의 최종 결과를 얻기 위한 값(ε=0.104471)을 사용하였다. 실험 결과에서 순차 검색과 S-ST는 모두 시퀀스 개수가 증가함에 따라 선형적으로 질의 처리 시간이 증가하고 있음을 알 수 있으나, 순차 검색 방식과 비교하여 S-ST 알고리즘은 시퀀스 개수의 변화에 거의 무관하게 실험 2와 같이 유사 허용치 값에 대한 동일한 성능 향상 결과를 보이고 있다.

실험 5: 그림 5.5는 시퀀스 평균 길이의 변화에 따른 질의 처리 시간의 비교 결과를 나타낸다. ε는 시퀀스 길이 160의 데이타에서 평균 약 100개의 최종 결과를 얻기 위한 값(ε= 0.137531)을 사용하였다. 순차 검색의 질의 처리 시간은 시퀀스 길이가 증가함에 따라 급격히 증가하고 있으나, S-ST는 완만한 증가 현상을 보이고 있다. 이 실험에서 순차 검색 방식과 비교하여 S-ST는 시퀀스 길이의 변화에 따라 약 25배에서 66배까지 나은 성능을 보이고 있다. 또한, 이러한 성능 개선 효과는 시퀀스의 길이가 길어질수록 커지는 양상을 보이고 있다.

## 6. 결 론

본 논문에서는 시계열 데이타베이스를 위한 모양 기반 검색 문제에 관하여 논의하였다. 모양 기반 검색이란 데이타베이스로부터 주어진 질의 시퀀스와 유사한 모양을 갖는 시퀀스를 검색하는 연산이다. 본 논문에서는 모양 기반 서브시퀀스 검색을 위한 새로운 유사 모델을 정의하고, 이를 지원하기 위한 인덱싱 및 질의 처리 방안에 관하여 논의하였다.

제안된 유사 모델에서는 기존의 값 기반 검색에서 사용하던 시프팅 변환, 스케일링 변환, 타임 워핑 변환, 이동 평균 변환 등 다양한 형태의 변환들의 조합을 동시에 지원함으로써 효과적인 모양 기반 검색을 가능하도록 한다. 기존의 연구에서 이러한 변환을 개별적으로 지원하는 기법들은 다수 제안된 바 있으나, 이들의 다양한 조합을 통하여 위와 같은 모양 기반 검색을 지원하는 유사 모델은 제안된 바 없다.

또한, 본 논문에서는 효율적인 동시에 착오 기각 없는 모양 기반 검색을 위하여 모든 추출 가능한 서브시퀀스들을 대상으로 디스크 기반 서브시퀀스 트리를 구성하는 기법과 이를 기반으로 질의를 효율적으로 처리하는 기법을 제안하였다.

제안된 방안의 우수성을 규명하기 위하여 실제 데이타인 S&P 500 주식 데이타를 이용한 다양한 실험들을 수행하였다. 실험 결과에 의하면 제안된 유사 모델을 지원하는 모양 기반 검색은 질의 시퀀스와 유사한 서브시퀀스들을 성공적으로 발견하였으며, 기존의 순차 검색 기법과 비교하여 최고 66배까지의 성능 개선 효과를 가지는 것으로 나타났다.

## 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] 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.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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, pp. 201-208, 1998.

[11] S. H. Park, W. W. Chu, J. H. Yoon, and C. Hsu, "Efficient Searches for Similar Subsequences of Difference Lengths in Sequence Databases," In *Proc. Int'l. Conf. on Data Engineering*, IEEE, pp. 23-32. 2000.

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

[13] C. S. Perng et al., "Landmarks: A New Model for Similarity-Based Pattern Querying in Time Series Databases," In *Proc. Int'l. Conf. on Data Engineering*, IEEE, pp. 33-42, 2000.

[14] M. Kendall, *Time-Series*, 2nd Edition, Charles Griffin and Company, 1979.

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

[16] 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.

[17] 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.

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

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

[20] S. H. Park, W. W. Chu, J. H. Yoon, and C. Hsu, A Suffix Tree for Fast Similarity Searches of Time-warped Subsequences in Sequence Databases, UCLA-CS-TR-990005, 1999.



### 윤 지 희

1982년 2월 한양대학교 전자공학과 졸업(학사). 1985년 3월 일본 구주대학교 정보공학과 졸업(석사). 1988년 3월 일본 구주대학교 정보공학과 졸업(박사). 1998년 3월 ~ 1999년 2월 UCLA대학교 전산학과 방문교수. 1988년 4월 ~ 현재 한림대학교 정보통신공학부 교수. 관심분야는 시계열 데이타베이스, 데이타 마이닝, XML, 공간 데이타베이스/GIS

### 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 방문연구원. 1994년 2월 ~ 1995년 2월 정보전자연구소 Post-Doc. 1999년 8월 ~ 2000년 8월 미국 IBM T.J. Watson Research Center Post-Doc. 1995년 3월 ~ 현재 강원대학교 컴퓨터정보통신공학부 교수. 관심분야는 데이타베이스 시스템, 데이타 마이닝, 멀티미디어 정보 검색, 공간 데이타베이스/GIS, 실시간 주기억장치 데이타베이스, 트랜잭션 관리

김 태 훈

2000년 2월 한림대학교 컴퓨터공학과 졸업(학사). 2002년 2월 한림대학교 컴퓨터공학과 졸업(석사). 2002년 3월 ~ 현재 (주)클래러스 연구원. 관심분야는 데이타베이스 시스템, 데이타 마이닝

박 상 현

1989년 2월 서울대학교 컴퓨터공학과 졸업(학사). 1991년 2월 서울대학교 컴퓨터공학과 졸업(석사). 2001년 1월 UCLA대학교 전산학과 졸업(박사). 1991년 3월 ~ 1996년 7월 대우통신 연구원. 2001년 2월 ~ 2002년 6월 IBM T.J. Watson Research Center Post-Doctorial Fellow. 2002년 8월 ~ 현재 포항공과대학교 컴퓨터공학과 교수. 관심분야 시공간 데이타베이스, 멀티미디어 데이타베이스, 데이타 마이닝, 데이타 웨어하우징, XML, 분산 데이타베이스