

시퀀스 데이터베이스에서 시간왜곡 변환에 기반하는 서브시퀀스 탐색

박 상 현*, 김 상 욱**, 노 용 기***

Subsequence Searching Under Time Warping in Sequence Databases

Sang-Hyun Park*, Sang-Wook Kim**, Woong-Kee Loh***

요약문

본 논문에서는 시퀀스 데이터베이스에서 시간왜곡 변환(time warping)을 지원하는 서브시퀀스 탐색 문제를 다룬다. 기존의 서브시퀀스 탐색 기법은 검색 성능이 데이터 시퀀스의 평균 길이의 이차 함수로 저하된다. 이러한 문제를 해결하기 위하여 본 논문에서는 세그먼트 기반 서브시퀀스 탐색 기법 (Segment-Based Approach for Subsequence Searches: SBASS)을 제안한다. SBASS는 데이터와 질의 시퀀스를 연속된 세그먼트들로 분할하여 다음의 두가지 조건을 만족하는 모든 데이터 시퀀스를 검색한다. (1) 세그먼트의 개수가 질의 시퀀스의 세그먼트 개수와 같다. (2) 모든 세그먼트 쌍 간의 거리가 주어진 오차 한도 이내이다. 제안된 세그먼트 분할 기법에서는 세그먼트가 서로 다른 길이를 갖도록 허용하며, 세그먼트 쌍 간의 유사성의 척도로서 시간왜곡 변환 거리를 이용한다. 효율적인 유사 서브시퀀스 탐색을 위하여, 각 데이터 세그먼트로부터 요소 값들이 단조적으로 변화하는 특성을 이용하여 특성 벡터를 추출하고, 추출된 특성 벡터를 이용하여 공간 인덱스를 생성한다. 질의는 이 인덱스를 이용하여 (1) R-트리 여과, (2) 특성 여과, (3) 순서 여과, (4) 후처리의 네 단계로 처리된다. 다양한 실험을 통하여 제안된 기법의 효율성을 입증한다.

1. 서론

시퀀스 데이터베이스에서의 효율적인 유사성 탐색에 관한 기존의 많은 연구들은 유사성 척도로서 유클리드 거리(Euclidean distance)를 사용한다 [1, 6, 7]. 그러나, 최근의 연구에서는 높은 계산 비용을 감수하더라도 보다 높은 정확성과 넓은 응용성을 얻기 위하여 시간왜곡 변환 거리(time warping distance)를 기반으로 하는 유사성 탐색 기법들이 널리 사용되고 있다 [9, 10, 11, 14]. 시간왜곡 변환(time warping)은 두 시퀀스 간의 차이를 최소화하기 위하여 한 시퀀스의 각 요소 값(element)이 다른 시퀀스의 하나 또는 인접한 여러 개의 요소 값들과 매칭하게 할 수 있도록 허용한다. 시간왜곡 변환은 음성, 오디오, 의료 정보를 위한 유사 검색에 널리 사용되어 왔다 [12].

시간왜곡 변환 거리는 전체 시퀀스 탐색(whole sequence search)과 서브시퀀스 탐색(subsequence search)에 모두 응용될 수 있다. 먼저, 전체 시퀀스 탐색에 대하여 계산 비용을 설명한다. 주어진 데이터 시퀀스 $\vec{X}$ 와 질의 시퀀스 $\vec{Q}$ 에 대하여, $|\vec{X}|$ 와 $|\vec{Q}|$ 를 각각 $\vec{X}$ 와 $\vec{Q}$ 의 길이라고 하자. 시간왜곡 변환 거리를 계산하는 시간은 $O(|\vec{X}||\vec{Q}|)$ 의 시간 복잡도를 갖는다 [2]. 평균 길이가 $\bar{L}$ 인 M 개의 데이터 시퀀스에 대하여 전체 시퀀스 탐색은 $O(M\bar{L}|\vec{Q}|)$ 의 복잡도를 갖는다. 따라서, 탐색 비용은 데이터 시퀀스의 전체 개수와 평균 길이 각각에 대하여 1차 함수로 증가한다.

다음은 서브시퀀스 탐색에 대하여 계산 비용을 설명한다. 길이

L 인 데이터 시퀀스 $\vec{X}$ 는 $L(L+1)/2$ 개의 서브시퀀스를 포함한다. 평균 길이가 $\bar{L}$ 인 M 개의 데이터 시퀀스에 대하여, 서브시퀀스 탐색하는 시간은 $O(M\bar{L}^2|\vec{Q}|)$ 의 시간 복잡도를 갖는다. 따라서, 탐색 비용은 데이터 시퀀스의 전체 개수에 대한 1차 함수로 증가하는 반면, 평균 길이에 대해서는 2차 함수로 증가한다.

시퀀스 탐색의 속도를 높이기 위하여 R-트리(R-tree) [8]와 같은 공간 인덱스 구조(spatial index structure)가 널리 이용된다. 그러나, 시간왜곡 변환 거리가 삼각 부등식(triangular inequality)을 만족하지 않기 때문에 [14], 삼각 부등식에 기반한 이러한 액세스 방법(access method)은 시간왜곡 변환 하에서 착오 기각(false dismissal) [1]을 유발한다. 이러한 문제를 극복하기 위하여, 참고문헌 [11]에서는 삼각 부등식에 기반하지 않는 분류된 접미사 트리(categorized suffix tree) [3, 13]를 이용한 서브시퀀스 탐색 기법을 제안하였다. 이 기법은 탐색 속도를 향상시켰으나, 탐색 시간은 여전히 데이터 시퀀스의 평균 길이에 2차 함수로 증가한다. 따라서, 많은 수의 긴 시퀀스가 저장되어 있는 데이터베이스 환경에서는 탐색 성능이 심각하게 저하된다.

본 논문에서는 위와 같은 문제를 해결하기 위한 새로운 방안을 제시한다. 본 논문의 일차적인 목적은 데이터 시퀀스의 평균 길이에 선형으로 비례하는 서브시퀀스 탐색 방법을 고안하는 것이다. 이를 위하여, 세그먼트 기반의 서브시퀀스 탐색 기법 (Segment-Based Approach for Subsequence Searches: SBASS)을 제안하고, SBASS를 위한 효율적인 인덱싱 방법을 제시한다.

SBASS는 데이터와 질의 시퀀스를 단조적으로 변화하는(monotonically changing) 연속된 요소 값들로 구성된 세그먼트 쌍으로 분할하고, 다음의 두가지 조건을 만족하는 모든 데이터 시퀀

* IBM T.J. Watson Research Center

** 강원대학교 컴퓨터정보통신공학부 부교수

*** 주식회사 Tmax Soft

스를 검색한다. (1) 세그먼트의 개수가 질의 시퀀스의 세그먼트 개수와 같다. (2) 모든 세그먼트 쌍 간의 거리가 오차 한도 (tolerance) 이내이다. 세그먼트의 길이는 다양하게 결정되므로, 시간왜곡 변환 거리가 세그먼트 쌍 간의 유사성 척도로 사용된다.

작은 기간 없이 유사한 서브시퀀스를 효율적으로 검색하기 위하여, 각 세그먼트로부터 특성 벡터(feature vector)를 추출하고, R-트리 필터(R-tree filter), 특성 필터(feature filter), 순서 필터(successor filter)의 세 필터를 이용하여 탐색을 수행한다. R-트리 필터는 세그먼트의 경계 값(boundary values)을 이용하여 질의 세그먼트와 유사한 후보 세그먼트 집합을 검색한다. 특성 필터는 나머지 특성들을 이용하여 이러한 후보 세그먼트 집합을 좀더 정제한다. 마지막으로, 순서 필터는 후보 세그먼트들 간의 순서 관계를 이용하여 후보 서브시퀀스를 반환한다.

2. 세그먼트 기반 서브시퀀스 탐색 기법 (SBASS)

본 절에서는 SBASS에 대하여 설명한다. SBASS는 질의 시퀀스와 비교 대상이 되는 서브시퀀스의 수를 줄이기 위하여 유사도를 계산할 때 타임 워핑 거리를 이용하는 기존의 방법을 약간 변형한다. SBASS는 먼저 데이터 시퀀스와 질의 시퀀스를 각각 세그먼트들의 순차적 리스트로 변환한다. 질의 시퀀스가 갖는 세그먼트의 수를 n 이라 가정하자. 질의 처리 시, SBASS는 n 개의 연속된 세그먼트들로 구성되는 정렬된 서브시퀀스(aligned subsequence)들만을 비교 대상으로 한다.

어떤 시퀀스 SQ 내의 정렬된 서브시퀀스 ASS란 다음과 같은 두 가지 조건을 만족하는 SQ 내의 서브시퀀스 SS 를 의미한다. (1) SS 의 첫 요소 값이 SQ 내의 한 세그먼트 S 의 첫 요소 값이어야 한다. (2) SS 의 마지막 요소 값이 S 혹은 SQ 내에서 S 이후에 나타나는 다른 세그먼트 S' 의 마지막 요소 값이어야 한다. 즉, 정렬된 서브시퀀스는 해당 시퀀스내의 세그먼트들의 리스트를 의미하므로, 세그먼트 중간에서 시작되거나 끝나는 서브시퀀스는 정렬된 서브시퀀스가 아니다. 따라서, N 개의 세그먼트들로 구성되는 데이터 시퀀스는 $N(N+1)/2$ 개의 정렬된 서브시퀀스를 갖는다.

SBASS에서는 질의 시퀀스 $\vec{q}$ 가 주어졌을 때, $\vec{q}$ 내의 세그먼트의 개수와 같은 수의 세그먼트를 갖는 정렬된 서브시퀀스만을 검색하여 $\vec{q}$ 와 비교한다. 정렬된 서브시퀀스 $\vec{x}$ 와 질의 시퀀스 $\vec{q}$ 가 모두 K 개의 세그먼트를 포함한다고 하자. $\vec{x}$ 와 $\vec{q}$ 간의 유사성은 $\vec{x}$ 와 $\vec{q}$ 의 대응되는 세그먼트 쌍 간의 유사성에 의해 결정된다. SBASS에서는 $\vec{x}$ 의 모든 i ($1 \leq i \leq K$)번째 세그먼트가 $\vec{q}$ 의 모든 i 번째 세그먼트와 유사할 때 $\vec{x}$ 와 $\vec{q}$ 가 유사하다고 판정한다. 표 1은 본 논문에서 사용된 표기법의 리스트를 보인다.

SBASS의 계산 비용은 다음과 같다. 세그먼트 내의 요소 값들의 평균 개수가 c 라면, 각 세그먼트 쌍의 시간왜곡 변환 거리를 계산하는 비용은 $O(c^2)$ 이다. 질의 시퀀스 $\vec{q}$ 내에는 $|\vec{q}|/c$ 개의 세그먼트가 존재한다. 따라서 $|\vec{q}|/c$ 개의 세그먼트를 갖는 정렬된 서브시퀀스와 질의 시퀀스간의 유사 정도를 계산하기 위한 비용은 $O(c^2 |\vec{q}|/c) = O(c |\vec{q}|)$ 이 된다. 전체 데이터시퀀스 $\vec{X}$ 내의 세그먼트의 수는 $|\vec{X}|/c$ 이므로, 이 데이터 시퀀스내에서 연속적으로 $|\vec{q}|/c$ 개의 세그먼트들을 갖는 정렬된 서브시퀀스들의 수는 $|\vec{X}|/c - |\vec{q}|/c + 1$ 이다.

따라서 $\vec{X}$ 와 $\vec{q}$ 간의 서브시퀀스 탐색을 위한 비용은 $(|\vec{X}|/c - |\vec{q}|/c + 1) \times (c|\vec{q}|) = |\vec{q}|(|\vec{X}| - |\vec{q}| + c)$ 가 된다. 이 비용은 데이터 시퀀스의 길이에 대한 1차 함수로 나타나므로, 2차 함수로 나타나는 기존의 기법에 비하여 좋은 성능을 얻을 수 있다.

2.1. 세그먼트 분할

하나의 시퀀스로부터 추출한 세그먼트의 순차적 리스트를 구하는

방법은 여러가지가 존재한다. 여기에서는 세그먼트로부터 유용한 특성 벡터를 추출하기 위해서 모든 세그먼트가 단조적으로 변화하는 패턴을 갖도록 하는 방법을 사용한다. 하나의 세그먼트 $\vec{a} = \langle a_1, a_2, \dots, a_N \rangle$ 에서 $a_1 \leq a_2 \leq \dots \leq a_N$ (단조 증가 패턴) 또는 $a_1 \geq a_2 \geq \dots \geq a_N$ (단조 감소 패턴) 성질이 만족되면, $\vec{a}$ 는 단조 변화 패턴을 갖는다고 한다. 그림 2는 시퀀스 $\vec{X}$ 로부터 세그먼트 표현 $\vec{X}^S$ 를 얻기 위한 알고리즘을 보인다. 여기에서, $\langle \rangle$ 은 빈(empty) 시퀀스를 표현하며, '·'은 피연산자 시퀀스들을 연결(concatenate)하는 이진 연산자이다. 예를 들어, $\langle 4,5,6 \rangle \cdot \langle 7,8 \rangle$ 의 결과는 $\langle 4,5,6,7,8 \rangle$ 이다.

표기법	설 명
$\vec{X}$	수치 값들로 구성된 시퀀스 $\vec{X} = \langle X_1, \dots, X_N \rangle$.
$\vec{x}$	서브시퀀스 또는 정렬된 서브시퀀스. $\vec{x} = \langle x_1, \dots, x_N \rangle$.
$\vec{X}^S$	$\vec{X}$ 의 세그먼트 표현. $\vec{X}^S[i]$ 는 $\vec{X}$ 의 i 번째 세그먼트.
$\vec{a}$	세그먼트. $\vec{a} = \langle a_1, \dots, a_N \rangle$
$F(\vec{a})$	$\vec{a}$ 로부터 얻어진 특성 벡터.
$IF(i)$	경향선(interpolation line)에서 얻어진 i 번째 요소 값.
$D(\vec{x}, \vec{y})$	두 정렬된 서브시퀀스 $\vec{x}$ 와 $\vec{y}$ 간의 거리 함수.
$D_w(\vec{a}, \vec{b})$	두 세그먼트 $\vec{a}$ 와 $\vec{b}$ 간의 시간왜곡 변환 거리 함수.
$D_H(F(\vec{a}), F(\vec{b}))$	두 특성 벡터 $F(\vec{a})$ 와 $F(\vec{b})$ 간의 거리 함수.

그림 1. 표기법 리스트.

```

입력: 시퀀스  $\vec{X}$ 
출력: 세그먼트 분할된 시퀀스  $\vec{X}^S$ 
 $\vec{X}^S \leftarrow \langle \rangle$ ;
 $\vec{T} \leftarrow \langle \rangle$ ;
for i  $\leftarrow$  1 to  $|\vec{X}|$  do
  if  $\vec{T} \cdot \langle X_i \rangle$  maintains the
    monotonically changing property
  then
     $\vec{T} \leftarrow \vec{T} \cdot \langle X_i \rangle$ ;
  else
    Insert  $\vec{T}$  into  $\vec{X}^S$ ;
     $\vec{T} \leftarrow \langle X_i \rangle$ ;
  endif
end for
Insert  $\vec{T}$  into  $\vec{X}^S$ ;
return  $\vec{X}^S$ ;

```

그림 2. 세그먼트 분할 알고리즘.

주어진 세그먼트 $\vec{a} = \langle a_1, \dots, a_N \rangle$ 에 대하여, 처음과 마지막 경계 요소 값 a_1 과 a_N 을 연결하는 선을 경향선(interpolation line)이라 정의한다. 만약 경향선으로부터 미리 정해진 한도값(threshold)을 초과하여 벗어나는 요소 값 a_1 가 존재하면 그 세그먼트를 둘로 분할할 수도 있다. 세그먼트 분할 과정은 모든 요소

값들이 한도값 이내에 들어올 때까지 재귀적으로(recursively) 진행할 수 있다.

얻어진 세그먼트의 개수는 시퀀스 내의 요소 값들의 분포에 의존적이나, 많은 경우 원래 시퀀스의 요소 값들의 개수에 비하여 매우 적다. 세그먼트 내의 요소 값들의 평균 개수를 나타내기 위하여 압축률(compaction ratio: CR), $CR = |\vec{X}| / |\vec{X}^S|$ 값을 이용한다.

2.2. 유사성 척도

주어진 두개의 시퀀스에 대한 적절한 유사성 척도를 찾기가 어려운 이유는 정량적으로 일치하는 시퀀스들이 정성적으로는 다를 수 있기 때문이다. SBASS에서는 사용자가 직감적으로 알 수 있고 서로 다른 길이의 시퀀스에 대해서도 적용 가능한 유사성 척도를 제안한다.

정의 1: K 개의 세그먼트를 포함하는 두개의 주어진 정렬된 서브시퀀스 $\vec{x}$ 와 $\vec{y}$ 에 대하여, 거리 함수 $D(\vec{x}, \vec{y})$ 는 다음과 같이 정의한다.

$$D(\vec{x}, \vec{y}) = \max(D_{mc}(\vec{x}^S[1], \vec{y}^S[1]), D_{mc}(\vec{x}^S[2], \vec{y}^S[2]), \dots, D_{mc}(\vec{x}^S[K], \vec{y}^S[K]))$$

여기에서, $D_{mc}(\vec{x}^S[i], \vec{y}^S[i])$ ($i = 1, 2, \dots, K$)는 두 세그먼트 $\vec{x}^S[i]$ 와 $\vec{y}^S[i]$ 에 대한 시간왜곡 변환 거리 함수이다. 이러한 정의는 만약 $D(\vec{x}, \vec{y}) = \epsilon$ 이면, 모든 세그먼트 쌍에 대하여 시간왜곡 변환 거리가 ϵ 이내임을 나타낸다.

□

시간왜곡 변환 거리 함수는 두 세그먼트 간의 거리를 최소화하기 위하여 한 세그먼트의 각 요소 값이 다른 세그먼트의 하나 이상의 인접한 요소 값들과 매칭되는 것을 허용한다 [2, 12]. 그 정의는 다음과 같다 [12].

정의 2: 주어진 두 개의 세그먼트 $\vec{a}$ 와 $\vec{b}$ 에 대하여, 시간왜곡 변환 거리 함수 $D_{mc}(\vec{a}, \vec{b})$ 는 다음과 같이 정의된다.

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

$$D_{mc}(\vec{a}, \langle \rangle) = D_{mc}(\langle \rangle, \vec{b}) = \infty$$

$$D_{mc}(\vec{a}, \vec{b}) = |\alpha_1 - \beta_1| + \min \begin{cases} D_{mc}(\vec{a}[2:], \vec{b}[2:]) \\ D_{mc}(\vec{a}[2:], \vec{b}) \\ D_{mc}(\vec{a}, \vec{b}[2:]) \end{cases}$$

여기에서, $\langle \rangle$ 는 빈 세그먼트를 나타내고, $\vec{a}[2:]$ 는 $\vec{a}$ 의 첫 요소 값을 제외한 나머지 모든 요소 값들을 포함하는 서브세그먼트(subsegment)이다.

3. 인덱스 생성

효율적인 서브시퀀스 탐색을 위하여, 데이터 세그먼트로부터 추출한 특성 벡터로 구성된 수정된 R-트리를 이용한다. 먼저 본 절에서 사용할 몇가지 표기법을 정의한다. 하나의 세그먼트 $\vec{a} = \langle a_1, a_2, \dots, a_N \rangle$ 내의 최소, 최대 요소 값은 각각 $\min(\vec{a})$, $\max(\vec{a})$ 로 표기한다. 단조 감소 패턴에서는 $\min(\vec{a}) = a_N$ 이고, $\max(\vec{a}) = a_1$ 이다.

$a_i - \min(\vec{a})$ 를 i번째 요소 값의 높이(height) h_i 라고 하면, $\vec{a}$ 는 $\vec{a} = \langle a_1, \min(\vec{a}) + h_2, \dots, \min(\vec{a}) + h_{N-1}, a_N \rangle$ 으로 다시 쓸 수 있다.

3.1. 특성 추출

주어진 세그먼트 $\vec{a} = \langle a_1, \dots, a_N \rangle$ 에 대하여, 단조 변화 특성을

이용하여 6-튜플 특성 벡터(6-tuple feature vector) $F(\vec{a}) = (B, L, N, H, Eu, Ed)$ 를 추출한다. B와 L은 각각 처음 요소 값($= a_1$)과 마지막 요소 값($= a_N$)이며, N은 요소 값의 개수이다. H는 두번째부터 (N-1)번째까지 요소 값들의 높이의 합이다. 즉, $H = \sum_{i=2}^{N-1} h_i = \sum_{i=2}^{N-1} (a_i - \min(\vec{a}))$ 이다. Eu는 $\vec{a}$ 의 경향선으로부터 벗어난 양의(non-negative) 편차 값 중의 최대값이며, Ed는 $\vec{a}$ 의 경향선으로부터 벗어난 음의(non-positive) 편차 값 중의 최소값이다.

세그먼트 $\vec{a}$ 의 경향선은 처음과 마지막 요소 값을 연결함으로써 얻어지므로, $IP(i) = \left(\frac{a_N - a_1}{N-1} \right) i + \left(a_1 - \frac{a_N - a_1}{N-1} \right)$ 과 같이 표현할 수 있다. a_1 과 a_N 을 각각 특성 벡터 내의 B와 L로 대체할 수 있으므로, 경향선은 $IP(i) = \left(\frac{L-B}{N-1} \right) i + \left(B - \frac{L-B}{N-1} \right)$ 와 같이 표현할 수도 있다. $\vec{a}$ 의 i번째 요소 값에 대한 편차 값(deviation value)은 $a_i - IP(i)$ 로 정의한다. 모든 요소 값에 대한 편차 값 중에서 가장 큰 양의 값이 Eu에, 가장 작은 음의 값이 Ed에 할당된다.

3.2. R-트리 생성

R-트리는 영역 질의와 점 질의를 모두 효율적으로 지원하는 균형(height-balanced) 공간 인덱스 구조이다 [8]. R-트리의 데이터 노드(leaf data node)에는 (MBR, ID) 형태의 엔트리가 포함되어 있다. 여기에서 ID는 공간 객체 식별자이고, MBR은 공간 객체의 최소 포함 사각형(minimum bounding rectangle, MBR)이다. 디렉토리 노드(non-leaf directory node)는 (MBR, ChildPointer) 형태의 엔트리를 포함한다. 여기에서 ChildPointer는 자식 노드(child node)의 주소이고 MBR은 자식 노드 내의 모든 MBR을 포함하는 MBR이다.

질의 세그먼트와 유사하지 않은 세그먼트들을 여과하기 위하여 데이터 세그먼트로부터 추출한 특성 벡터들을 이용하여 R-트리를 생성한다. 각 특성 벡터는 데이터 노드의 한 엔트리에 해당한다. 효율적인 여과를 위하여 R-트리 구조를 수정한다. 수정된 R-트리는 특성 벡터 $F(\vec{a}) = (B, L, N, H, Eu, Ed)$ 내의 처음과 마지막 요소 값(B와 L)만을 키 애트리뷰트(key attribute)로 이용한다. 나머지 네 개의 특성 값들은 이후의 여과 과정을 위하여 데이터 노드에만 저장된다. 따라서, 데이터 노드 엔트리의 구조는 (MBR, ID, OtherFeatures)로 되며, 여기에서 MBR은 2차원의 점 (B, L), ID는 인덱싱된 세그먼트의 식별자, OtherFeatures는 나머지 특성 값들(N, H, Eu, Ed)이다. 데이터 노드 엔트리의 구조는 변경되지 않는다.

데이터베이스 내에서 실제 데이터 시퀀스를 효율적으로 찾을 수 있고 세그먼트들 간의 순서 관계를 쉽게 알 수 있도록 세그먼트 식별자는 (sequence#, segment#)로 표현한다. 세그먼트 $\vec{a}$ 가 식별자 (t, s)를 갖는다면, 바로 이전에 오는 세그먼트 $prev(\vec{a})$ 는 식별자 (t, s-1)를 갖고, 바로 다음의 세그먼트 $next(\vec{a})$ 는 식별자 (t, s+1)를 갖는다.

4. 질의 처리

본 절에서는 질의 시퀀스 $\vec{q}$ 와 오차 한도 ϵ 이내의 정렬된 서브시퀀스를 효율적으로 검색하기 위한 질의 처리 알고리즘을 제시한다. $\vec{x}$ 와 $\vec{q}$ 간의 거리가 ϵ 이내인 경우는 모든 세그먼트 쌍이 ϵ 이내의 시간왜곡 변환 거리를 갖는 경우이다. 제안된 알고리즘은 그림 3에서 보인 바와 같이 세 가지의 여과 과정과 후처리 과정으로 구성된다.

제안된 알고리즘은 먼저 질의 시퀀스 $\vec{q}$ 를 세그먼트 분할한 $\vec{q}^S$ 로 변환한다. 그리고 나서, 각 질의 세그먼트는 R-트리 필터로 보내진다. 세그먼트의 처음과 마지막 값을 이용하여 R-트리 필터는 전달된 질의 세그먼트와 유사한 후보 세그먼트의 집합을 반환한다.

R-트리 필터는 환경에 따라 순차적 또는 병렬적으로 실행할 수 있다. 특성 필터는 R-트리 데이터 노드에 포함된 모든 특성들을 이용하여 R-트리 필터로부터의 출력을 좀더 정제한다. 순서 필터는 모든 특성 필터로부터의 출력을 받아서 후보 세그먼트 간의 순서 관계를 이용하여 후보 서브시퀀스를 조립한다. 후처리 과정에서 실제 데이터 시퀀스 $\vec{x}$ 를 데이터베이스에서 읽어 들여 제안된 유사성 척도 $D(\vec{x}, \vec{q})$ 를 적용하여 착오 채택(false alarm) [1]을 제거하여 최종 결과를 얻는다.

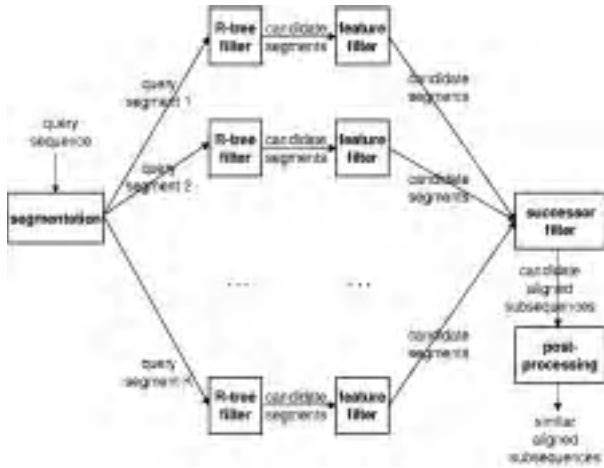


그림 3. 질의 처리 과정.

4.1. R-트리 필터

주어진 질의 세그먼트의 처음과 마지막 요소 값에 대응되는 점 (B, L) 과 오차 한도 ϵ 에 대하여, R-트리 필터는 2-차원 질의 사각형 $[B - \epsilon, B + \epsilon], [L - \epsilon, L + \epsilon]$ 내에 위치하는 데이터 점들을 찾는다. 질의 사각형에 속하는 점들의 집합은 처음과 마지막 요소 값들이 각각 $[B - \epsilon, B + \epsilon]$ 과 $[L - \epsilon, L + \epsilon]$ 범위 안에 포함되는 데이터 세그먼트들의 집합을 나타낸다. 다음의 정리를 통하여 질의 사각형 외부의 데이터 세그먼트는 항상 질의 세그먼트로부터 ϵ 보다 큰 시간왜곡 변환 거리를 가짐을 보인다.

정리 1: 주어진 세그먼트 $\vec{a}$ 와 $\vec{b}$ 가 각각 데이터 점 (B, L) 과 (B', L') 으로 나타낼 때, 만약 $|B - B'| > \epsilon$ 또는 $|L - L'| > \epsilon$ 이면, 항상 $D_{tc}(\vec{a}, \vec{b}) > \epsilon$ 이다.

증명 : 생략

□

R-트리의 데이터 노트 엔트리가 세그먼트의 여섯 개의 특성 값을 모두 포함하므로, R-트리 필터는 레코드 $(ID(\vec{a}), F(\vec{a}))$ 의 집합을 반환한다. 여기에서, $ID(\vec{a})$ 와 $F(\vec{a})$ 는 각각 $\vec{a}$ 의 식별자와 특성 벡터이다.

R-트리 필터의 여과율(filtering rate)은 질의 시에 주어진 오차 한도 ϵ 에 따라 좌우된다. ϵ 이 증가하여 질의 결과 개수가 증가하면, 여과율은 낮아진다. 여과율은 세그먼트의 데이터 분포에 따라서도 좌우된다. 질의 사각형이 데이터 밀도가 낮은 영역에 떨어지면 여과율은 높아지고, 반면에 데이터 밀도가 높은 영역에서는 여과율은 감소한다.

4.2. 특성 필터

특성 필터는 R-트리 필터의 출력에 대하여, 모든 추출된 특성 값들을 이용하여 좀더 정확하게 두 세그먼트 간의 거리를 추정함으로써 필터링을 수행한다. 특성 필터에 사용되는 거리 함수를 설명하

기 전에 기본적인 표기법과 개념을 설명한다.

4.2.1. 요소 값들의 범위

주어진 세그먼트에 대하여, 그 특성 벡터는 쉽게 얻어진다. 그러나, 특성 벡터로부터 세그먼트에 포함된 정확한 요소 값들은 구할 수 없다. 대신에, 각 요소 값들의 가능한 범위를 추정할 수 있다. 경향선 $IP(i)$ 와 최대 편차 값 Eu , 최소 편차 값 Ed 에 대하여, i 번째 요소 값이 $IP(i) + Ed$ 와 $IP(i) + Eu$ 사이에 존재함은 명백하다. i 번째 요소 값의 범위는 그 값이 세그먼트의 최소값보다 크고 최대값보다 작다는 분명한 사실을 이용하여 좀더 좁아질 수 있다. 즉, i 번째 요소 값은 $\max(IP(i) + Ed, \min(\vec{a}))$ 와 $\min(IP(i) + Eu, \max(\vec{a}))$ 사이에 존재한다. i 번째 요소 값의 범위를 나타내기 위하여 LB_i 와 UB_i 표기법을 사용한다.

정의 3: 주어진 특성 벡터 $F(\vec{a}) = (B, L, N, H, Eu, Ed)$ 에 대하여, i 번째 요소 값의 하한과 상한 값은 다음과 같이 정의한다.

$$LB_i = \max(IP(i) + Ed, \min(\vec{a}))$$

$$UB_i = \min(IP(i) + Eu, \max(\vec{a}))$$

□

4.2.2. UBset과 LBset

주어진 특성 벡터 $F(\vec{a})$ 와 $\min(\vec{a})$ 와 $\max(\vec{a})$ 사이의 값 v 에 대하여, 세그먼트 $\vec{a}$ 를 묘사하기 위해 유용한 두 가지 개념을 소개한다.

- $UBset(F(\vec{a}), v)$ 은 v 보다 작은 UB_i 의 집합이다.

$$UBset(F(\vec{a}), v) = \{UB_i | UB_i < v\}.$$

- $LBset(F(\vec{a}), v)$ 은 v 보다 큰 LB_i 의 집합이다.

$$LBset(F(\vec{a}), v) = \{LB_i | LB_i > v\}.$$

특성 벡터로부터 모든 i 번째 요소 값에 대한 UB_i 와 LB_i 를 구함으로써 $UBset(F(\vec{a}), v)$ 와 $LBset(F(\vec{a}), v)$ 를 쉽게 계산할 수 있다. UB_i 와 LB_i 모두 0(1)에 얻어지므로, $UBset(F(\vec{a}), v)$ 와 $LBset(F(\vec{a}), v)$ 는 $O(|\vec{a}|)$ 의 계산 시간이 필요하다.

$UBset(F(\vec{a}), v)$ 내의 모든 값들 중에서 가장 큰 값인 $\max(UBset(F(\vec{a}), v))$ 가 v 에 가장 가깝다. $\max(UBset(F(\vec{a}), v))$ 가 UB_p 와 같아지는 경우는 (1) $UB_p < v$ 이고 $UB_{p+1} \geq v$ ($\vec{a}$ 가 증가하는 패턴을 가질 때) 또는 (2) $UB_p < v$ 이고 $UB_{p-1} \geq v$ ($\vec{a}$ 가 감소하는 패턴을 가질 때)인 경우이다. 이러한 요소 값의 위치 p 는 다음의 공식에 의하여 구해진다.

$$p = \begin{cases} \left\lceil \left[\left(\left(\frac{N-1}{L-B} \right) (v - Eu - B) + \frac{L-B}{N-1} \right) - 1 \right] \right\rceil & (\vec{a} \text{가 증가하는 패턴을 가질 때}) \\ \left\lfloor \left[\left(\left(\frac{N-1}{L-B} \right) (v - Eu - B) + \frac{L-B}{N-1} \right) + 1 \right] \right\rfloor & (\vec{a} \text{가 감소하는 패턴을 가질 때}) \end{cases}$$

여기에서, $[arg]$ 는 arg 보다 작지 않은 최소 정수를 반환하며, $\lceil arg \rceil$ 는 arg 보다 크지 않은 최대 정수를 반환하는 함수이다.

유사하게, $LBset(F(\vec{a}), v)$ 내의 모든 값들 중에서 가장 작은 값인 $\min(LBset(F(\vec{a}), v))$ 가 v 에 가장 가깝다.

$\min(LBset(F(\vec{a}), v))$ 가 LB_q 와 같아지는 경우는 (1) $LB_q > v$ 이고 $LB_{q-1} \leq v$ ($\vec{a}$ 가 증가하는 패턴을 가질 때) 또는 (2) $LB_q > v$

이고 $LB_{q+1} \leq v$ ($\vec{a}$ 가 감소하는 패턴을 가질 때)인 경우이다. 이러한 요소 값의 위치 q 는 다음의 공식에 의하여 구해진다.

$$q = \begin{cases} \lfloor \left((-\frac{N-1}{L-B})(v - Ed - B + \frac{L-B}{N-1}) + 1 \right) \rfloor & (\vec{a} \text{가 증가하는 패턴을 가질 때}) \\ \lceil \left((-\frac{N-1}{L-B})(v - Ed - B + \frac{L-B}{N-1}) - 1 \right) \rceil & (\vec{a} \text{가 감소하는 패턴을 가질 때}) \end{cases}$$

두 요소 값의 위치 모두 0(1)에 계산되므로, $\max(UBset(F(\vec{a}), v))$ 와 $\min(LBset(F(\vec{a}), v))$ 의 계산도 0(1)이다.

조건 $UB_p = \max(UBset(F(\vec{a}), v))$ 를 만족하는 요소 값의 위치 p 가 정해지면, $UBset(F(\vec{a}), v)$ 내의 요소 값들의 개수는 쉽게 구해진다. $\vec{a}$ 가 증가하는 패턴을 가질 때, 모든 i ($1 \leq i \leq p$)에 대하여 $UB_i < v$ 가 성립한다. 따라서, $|UBset(F(\vec{a}), v)| = p$ 이다. $\vec{a}$ 가 감소하는 패턴을 가질 때, 모든 i ($p \leq i \leq |\vec{a}|$)에 대하여 $UB_i < v$ 가 성립한다. 그러므로, $|UBset(F(\vec{a}), v)| = |\vec{a}| - p + 1$ 이다. 같은 방식으로, $LBset(F(\vec{a}), v)$ 내의 요소 값의 개수는 조건 $LB_q = \min(LBset(F(\vec{a}), v))$ 를 만족하는 요소 값의 위치 q 로부터 구해진다. $\vec{a}$ 가 증가하는 패턴을 가지면 $|LBset(F(\vec{a}), v)| = |\vec{a}| - q + 1$ 이고, 감소하는 패턴을 가지면 $|LBset(F(\vec{a}), v)| = q$ 이다. (그림 4 참조)

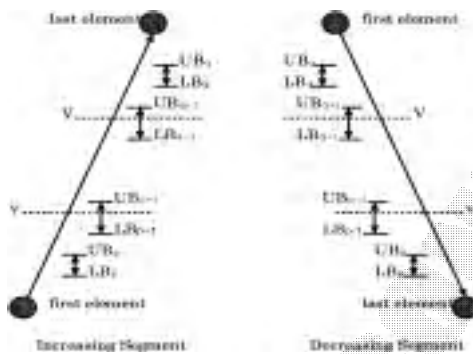


그림 4. 조건 $UB_p = \max(UBset(F(\vec{a}), v))$ 를 만족하는 요소 값의 위치 p 와 $LB_q = \min(LBset(F(\vec{a}), v))$ 를 만족하는 요소 값의 위치 q .

4.2.3. 특성 벡터 거리 함수

여기에서는 두 특성 벡터 $F(\vec{a})$ 와 $F(\vec{\beta})$ 에 대한 거리 함수 $D_H()$ 를 정의한다. 이 함수 값은 항상 $D_{uc}(\vec{a}, \vec{\beta})$ 보다 작다. 설명의 편의를 위하여 $\max(\vec{a}) \geq \max(\vec{\beta})$ 라고 가정한다. 만약 성립되지 않는다면, $\vec{a}$ 와 $\vec{\beta}$ 를 교환하면 된다. 거리 함수는 비교 대상인 두 특성 벡터의 범위에 따라 아래와 같이 달라진다.

정의 4: 주어진 $\vec{a}, \vec{\beta}$, $F(\vec{a}) = (B, L, N, H, Eu, Ea)$, $F(\vec{\beta}) = (B', L', N', H', Eu', Ea')$ 에 대하여, 거리 함수 $D_H(F(\vec{a}), F(\vec{\beta}))$ 는 다음과 같이 정의한다.

- 경우 1: $\vec{a}$ 와 $\vec{\beta}$ 가 서로 만나지 않을 때 ($\min(\vec{a}) > \max(\vec{\beta})$),

$$D_H(F(\vec{a}), F(\vec{\beta}))$$

$$= |B - B'| + |L - L'| + \max \left\{ \begin{aligned} & (N-2)(\min(\vec{a}) - \max(\vec{\beta})) + H \\ & (N'-2)(\min(\vec{a}) - \min(\vec{\beta})) - H' \end{aligned} \right.$$

- 경우 2: $\vec{a}$ 와 $\vec{\beta}$ 가 서로 겹칠 때 ($\min(\vec{\beta}) \leq \min(\vec{a}) \leq \max(\vec{\beta})$),

$$D_H(F(\vec{a}), F(\vec{\beta}))$$

$$= |B - B'| + |L - L'| +$$

$$(|LBset(F(\vec{a}), \max(\vec{\beta}))| - 1) \\ (\min(LBset(F(\vec{a}), \max(\vec{\beta}))) - \max(\vec{\beta})) +$$

$$(|UBset(F(\vec{\beta}), \min(\vec{a}))| - 1) \\ (\min(\vec{a}) - \max(UBset(F(\vec{\beta}), \min(\vec{a}))))$$

- 경우 3: $\vec{a}$ 가 $\vec{\beta}$ 를 포함할 때 ($\min(\vec{a}) < \min(\vec{\beta})$),

$$D_H(F(\vec{a}), F(\vec{\beta}))$$

$$= |B - B'| + |L - L'| +$$

$$(|LBset(F(\vec{a}), \max(\vec{\beta}))| - 1) \\ (\min(LBset(F(\vec{a}), \max(\vec{\beta}))) - \max(\vec{\beta})) +$$

$$(|UBset(F(\vec{a}), \min(\vec{\beta}))| - 1) \\ (\min(\vec{\beta}) - \max(UBset(F(\vec{a}), \min(\vec{\beta}))))$$

□

$\vec{\beta}$ 를 질의 세그먼트라고 하면, R-트리 필터로부터 반환된 레코드 $(ID(\vec{a}), F(\vec{a}))$ 에 대하여 거리 함수 $D_H(F(\vec{a}), F(\vec{\beta}))$ 가 적용된다. 만약 $D_H(F(\vec{a}), F(\vec{\beta})) > \epsilon$ 이면, 레코드 $(ID(\vec{a}), F(\vec{a}))$ 는 다음의 정리에 의하여 여과되어 버려진다.

정리 2: 주어진 $\vec{a}, \vec{\beta}$, $F(\vec{a}), F(\vec{\beta})$ 에 대하여, 만약 $D_H(F(\vec{a}), F(\vec{\beta})) > \epsilon$ 이면, $D_{uc}(\vec{a}, \vec{\beta}) > \epsilon$ 이다.

증명: 생략.

□

$D_H(F(\vec{a}), F(\vec{\beta}))$ 가 계산 복잡도 0(1)을 가지므로, R-트리 필터를 통해 반환된 레코드의 개수를 N 이라 하면, 특성 필터는 $O(N)$ 의 복잡도를 갖는다. $D_H(F(\vec{a}), F(\vec{\beta})) \geq |B - B'| + |L - L'|$ 이므로 특성 벡터는 여과율을 높인다. 특성 필터로부터의 결과로 후보 세그먼트들을 얻는다.

5. 순서 필터

순서 필터는 특성 필터로부터 반환된 후보 세그먼트들 간의 순서 관계를 이용하여 그들을 연결하고, 정렬된 후보 서브시퀀스의 집합을 생성한다.

이 과정에서, 두 개의 후보 세그먼트 $\vec{a}$ 와 $\vec{\beta}$ 를 연결하여 하나의 서브시퀀스 $\vec{a} \cdot \vec{\beta}$ 를 만들기 위해서는 다음의 두가지 조건이 만족되어야 한다. (1) $\vec{\beta} = next(\vec{a})$, 그리고 (2) $\vec{a}$ 가 i 번째 특성 필터로부터의 결과에 포함되어 있다면, $\vec{\beta}$ 는 $(i + 1)$ 번째 특성 필터의 결과에 포함되어 있어야 한다. 이러한 연결 과정은 첫번째 특성 필터의 출력으로부터 시작하여 마지막 특성 필터의 출력에서 종료한다. 순서 필터에서 생성된 세그먼트들의 순서 집합은 정렬된 후보 서브시퀀스라고 부른다. 후보 서브시퀀스를 형성하지 못한 후보 세그먼트들은 순서 필터에 의하여 여과되어 버려진다.

마지막으로, 후처리 과정에서는 (1) 순서 필터로부터 정렬된 후보 서브시퀀스 $\vec{x}$ 를 얻고, (2) 데이터베이스로부터 $\vec{x}$ 를 포함하는 데이터 시퀀스를 액세스하고, (3) 유사성 척도 $D(\vec{x}, \vec{q})$ 를 적용함으로써 남아있는 착오 채택을 최종적으로 제거한다.

6. 성능 평가

본 절의 목적은 제안된 SBASS가 (1) 데이터 시퀀스의 전체 개수와 평균 길이 모두에 대하여 선형으로 비례하는 성능을 가지며, (2) 순차 검색 알고리즘에 비하여 얼마나 성능 개선 효과를 가지는지 규명하는 것이다.

6.1. SBASS의 성능

제 3 절에서 SBASS가 데이터 시퀀스의 전체 개수와 평균 길이 모두에 대하여 선형의 계산 비용을 갖는다고 하였다. 이를 증명하기 위하여, SBASS와 순차 검색 알고리즘을 구현하고, 랜덤워크 데이터 시퀀스를 이용하여 성능을 측정하였다. 데이터 시퀀스를 생성하기 위한 공식은 다음과 같다.

$$\vec{X}[0] = \text{rand}([10, 100])$$

$$\vec{X}[i] = \vec{X}[i-1] + \text{rand}([-10, 10])$$

먼저 데이터 시퀀스의 평균 길이를 500으로 고정하고, 데이터 시퀀스의 개수를 1,000에서 5,000 개로 증가시켰다. 다음에, 데이터 시퀀스의 개수를 500으로 고정하고 데이터 시퀀스의 평균 길이를 1,000에서 5,000 개로 증가시켰다. 질의 시퀀스도 데이터 시퀀스와 같은 방법으로 생성하였다. 질의 시퀀스의 평균 길이는 데이터 시퀀스의 10 분의 1로, 오차 한도 ϵ 은 10^{-2} %의 결과를 얻도록 정하였다.

그림 5는 데이터 시퀀스의 개수가 증가함에 따른 SBASS의 질의 처리 시간을 나타낸 것이다. 실험 결과에 의하면, SBASS의 질의 처리 시간은 데이터 시퀀스의 개수에 대한 1차 함수의 형태로 증가하는 것으로 나타났다.

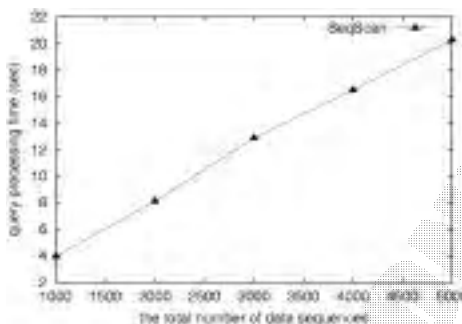


그림 5. 데이터 시퀀스의 개수가 증가함에 따른 SBASS의 질의 처리 시간(데이터 시퀀스 평균 길이 = 500).

그림 6은 데이터 시퀀스의 평균 길이가 증가함에 따른 SBASS의 질의 처리 시간을 나타낸 것이다. 실험 결과에 의하면, SBASS의 질의 처리 시간은 데이터 시퀀스의 평균 길이에 대한 1차 함수에 가까운 형태로 증가하는 것으로 나타났다.

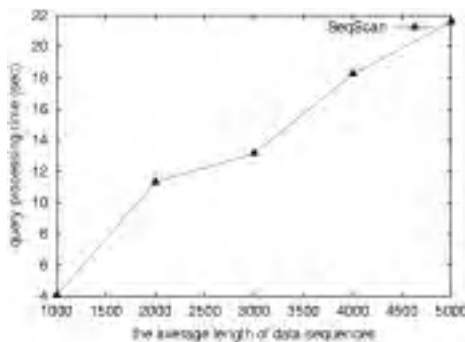


그림 6. 데이터 시퀀스의 평균 길이가 증가함에 따른 SBASS의 질의 처리 시간(데이터 시퀀스 전체 개수 = 500).

6.2. 성능 비교

제안된 기법의 성능 개선 효과를 규명하기 위하여 순차 검색과 질의 처리 성능을 비교 검토하였다. 성능 비교 실험을 위하여 UC Irvine KDD Archive(<http://kdd.ics.uci.edu>)에서 얻어진 "Pseudo Periodic Synthetic Time Series" 데이터 집합을 이용하였다. "Pseudo Periodic Synthetic Time Series"는 시계열 데이터베이스에서 시퀀스 매칭 방법들의 성능을 검증하기 위하여 다음의 함수를 이용하여 생성된 데이터 집합이다.

$$\vec{X} = \sum_{i=3}^7 \frac{1}{2^i} \sin(2\pi(2^{2+i} + \text{rand}(2^i))\vec{t}), \text{여기에서, } 0 \leq \vec{t} \leq 1 \text{이다.}$$

길이가 모두 10,000인 100 개의 데이터 시퀀스들을 생성하였다. 또한, 질의 처리를 위하여 같은 함수를 이용하여 평균 길이가 1,000인 질의 시퀀스를 생성하였다. 각각의 질의 시퀀스에 대하여 1에서 30까지의 서로 다른 오차 한도를 주어 SBASS와 순차 검색으로 각각 처리하도록 한 다음, 각각에 대하여 수행 시간의 평균을 측정하였다.

그림 7은 실험 결과를 나타낸 것이다. 제안된 기법은 질의의 오차 한도에 관계 없이 항상 순차 검색보다 나은 성능을 보이는 것으로 나타났으며, 최대 약 5배까지 향상된 성능을 보였다. 또한, 오차 한도가 작을수록 성능 개선의 효과는 점점 커지는 것으로 나타났다.

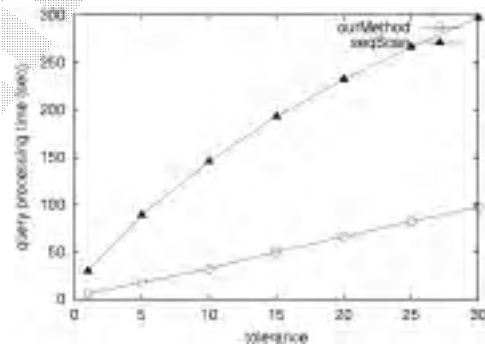


그림 7. 오차 한도 값을 증가시키에 따른 제안된 기법과 순차 검색의 질의 처리 시간.

7. 결론

시간왜곡 변환 거리는 데이터 시퀀스에 대한 유용한 유사성 척도이나 많은 계산 비용을 요구한다. 특히, 서브시퀀스 탐색에서는 탐색 비용이 데이터 시퀀스의 길이에 대한 2차 함수로 증가하므로, 데이터 시퀀스의 길이가 길어지면 성능이 심각하게 저하되는 문제가 있다.

이러한 문제를 해결하기 위하여, 본 논문에서는 세그먼트 기반의 서브시퀀스 탐색 기법(SBASS)을 제안하였다. 제안된 기법은 질의 시퀀스와 서브시퀀스를 세그먼트 단위로 분할하고 비교한다. 착오 기각 없이 빠른 유사 시퀀스 검색을 위하여, R-트리 필터, 특성 필터, 순서 필터를 이용한 새로운 탐색 방법을 제안하였다. 실험을 통한 성능 평가로 제안된 기법의 효율성을 보였다.

본 논문의 공헌은 다음과 같다. (1) SBASS와 유사성 척도를 제안하였다. (2) 세그먼트를 정확하게 표현하기 위한 특성 벡터의 추출 방법을 제시하였다. (3) 세그먼트의 단조 변화 패턴을 이용하여 상수 시간 복잡도의 거리 함수를 유도하였다. (4) 질의 처리 속도를 높이기 위하여 세 가지 여과 과정을 제안하였다.

Acknowledgement

본 연구는 한국과학재단의 99 해외 Post-Doc 방문 프로그램 및 2000 목적기초연구 중 지역대학 우수과학자 지원 프로그램(과제 번호: 2000-1-51200-006-1)과 강원대학교 멀티미디어 연구센터를 통한 정보통신 우수시범학교 지원사업의 연구비 지원을 받았습니다.

참고 문헌

- [1] R. Agrawal, C. Faloutsos, A. Swami, "Efficient Similarity Search in Sequence Databases," Proc. FODO, pp. 69-84, 1993.
- [2] D. J. Berndt, J. Clifford, "Finding Patterns in Time Series: A Dynamic Programming Approach," Advances in Knowledge Discovery and Data Mining, AAAI/MIT, pp. 229-248, 1996.
- [3] P. Bieganski, J. Riedl, J. V. Carlis, "Generalized Suffix Trees for Biological Sequence Data: Applications and Implementation," Proc. Hawaii Int'l Conf. on System Sciences, 1994.
- [4] T. Bozkaya, N. Yazdani, M. Özsoyoğlu, "Matching and Indexing Sequences of Different Lengths," Proc. ACM CIKM, pp. 128-135, 1997.
- [5] C. Faloutsos, K. Lin, "FastMap: A Fast Algorithm for Indexing, Data-Mining and Visualization of Traditional and Multimedia Datasets," Proc. ACM SIGMOD, pp. 163-174, 1995.
- [6] C. Faloutsos, M. Ranganathan, Y. Manolopoulos, "Fast Subsequence Matching in Time-Series Databases," Proc. ACM SIGMOD, pp. 419-429, 1994.
- [7] D. Q. Goldin, P. C. Kanellakis, "On Similarity Queries for Time-Series Data: Constraint Specification and Implementation," Proc. Constraint Programming, pp. 137-153, 1995.
- [8] A. Guttman, "R-trees: A Dynamic Index Structure for Spatial Searching," Proc. ACM SIGMOD, pp. 47-57, 1984.
- [9] E. J. Keogh, M. J. Pazzani, "Scaling up Dynamic Time Warping to Massive Datasets," Proc. Principles and Practice of Knowledge Discovery in Databases, 1999.
- [10] S. Park, D. Lee, W. W. Chu, "Fast Retrieval of Similar Subsequences in Long Sequence Databases," Proc. 3rd IEEE KDEX, pp. 60-67, 1999.
- [11] S. Park, W. W. Chu, J. Yoon, C. Hsu, "Efficient Searches for Similar Subsequences of Different Lengths in Sequence Databases," Proc. IEEE ICDE, pp. 23-32, 2000.
- [12] L. Rabiner, B.-H. Juang, Fundamentals of Speech Recognition, Prentice Hall, 1993.
- [13] G. A. Stephen, String Searching Algorithms, World Scientific Publishing, 1994.
- [14] B.-K. Yi, H. V. Jagadish, C. Faloutsos, "Efficient Retrieval of Similar Time Sequences Under Time Warping," Proc. IEEE ICDE, pp. 201-208,