시계열 서브시퀀스 매칭을 위한 최적의 다중 인덱스 구성 방안

임승환°, 박희진, 김상욱
한양대학교 정보통신공학과
{shlim°, hjpark, wook}@hanyang.ac.kr

Optimal Construction of Multiple Indexes for Time-Series Subsequence Matching

Seung-Hwan Lim°, Hee-Jin Park, Sang-Wook Kim
College of Info. & Comm., Hanyang University

요 약

서브시퀀스 매칭은 주어진 질의 시퀀스와 변화의 추세가 유사한 서브시퀀스들을 시계열 데이터베이스로부터 검색하는 연산이다. 본 논문에서는 윈도우 크기 효과로 인한 서브시퀀스 매칭의 심각한 성능 저하 현상을 정량적으로 관찰하여, 하나의 윈도우 크기를 대상으로 만든 단 하나의 인덱스만을 이용하는 것은 실제 응용에서 만족할만한 성능을 제공할 수 없다는 것을 규명하였다. 또한, 이러한 문제로 인해 다양한 윈도우 크기들을 기반으로 다수의 인덱스들을 구성하여 서브시퀀스 매칭을 수행하는 인덱스 보간법의 응용이 필요함을 보였다. 인덱스 보간법을 응용하여 서브시퀀스 매칭을 수행하기 위해서는 먼저 다수의 인덱스들을 위한 윈도우 크기들을 결정해야 한다. 본 연구에서는 물리적 데이터베이스 설계 방식을 이용하여 이러한 최적의 다수의 윈도우 크기들을 선정하는 문제를 해결하였다. 이를 위하여 시계열 데이터베이스에서 수행될 예정인 질의 시퀀스들의 집합과 인덱스 구성의 기반이 되는 윈도우들의 크기의 집합이 주어질 때, 전체 서브시퀀스 매칭들을 수행하는 데에 소요되는 비용을 예측할 수 있는 공식을 산출하였다. 또한, 이 비용 공식을 이용하여 전체 서브시퀀스 매칭들의 성능을 극대화 할 수 있는 최적의 윈도우 크기들을 결정하는 알고리즘을 제안하였으며, 이 알고리즘의 최적성과 효율성을 이론적으로 규명하였다. 끝으로, 실험에 의한 성능 평가를 통하여 제안된 기법의 우수성을 제시하였다.

1. 서론

우리 주위에는 주가지수, 환율, 기온과 같이 시간에 따라 값이 변하는 객체들이 존재한다. 이러한 각 객체의 일정 기간 동안 변화한 값들을 기록한 것을 그 객체에 대한 시계열 데이터 시퀀스(data sequence)라고 부른다[Agr93][Fal94]. 시계열 데이터 시퀀스들이 저장되어 있는 데이터베이스를 시계열 데이터베이스(time-series database)라 한다[Agr93][Fal94][Moo01]. 유사 서브시퀀스 매칭은 데이터베이스 D 내에 존재하는 시퀀스 $S_1, S_2, ..., S_N$에 대하여 질의 시퀀스 Q와 유사한 서브시퀀스 X를 포함하는 시퀀스 S_i와 이 서브시퀀스 X가 S_i내에서 시작하는 위치를 검색한다. 이때, $S_1, S_2, ..., S_N$ 및 Q는 서로 다른 길이를 갖는 것이 허용된다[Agr93][Fal94][Moo01]. 유사 시퀀스 매칭은 데이터 마이닝(data mining) 및 데이터 웨어하우징(data warehousing) 분야에서 중요한 연산으로 사용된다[Raf97].

두 시퀀스 간에 유사성을 측정하는 척도로는 아래와 같이 정의되는 유클리드 거리(Euclidean distance)가 널리 사용된다[Agr93][Cha99][Fal94][Raf97][Raf99]. 즉, 길이가 n인 서로 다른 두 시퀀스 $X(=<x_0, x_1, ..., x_{n-1}>)$와 $Y(=<y_0, y_1, ..., y_{n-1}>)$를 각각 n차원 공간상의 한 점으로 매핑하고, 두 점 사이의 유클리드 거리 D(X, Y)가 ε 이하이면, 두 시퀀스 X, Y는 ε-매치(ε-match)한다고 한다[Moo01].

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

서브시퀀스 매칭을 위한 처리 기법으로 대표적인 두가지 방법이 있다. 이들은 각각 참고문헌 [Fal94]와 [Moo01]에서 소개된 방법이다. 참고 문헌 [Moo01]의 명칭을 따라 본 논문에서는 [Fal94]의 기법을 FRM, [Moo01]의 기법을 Dual-Match라 부른다. FRM과 Dual-Match는 효과적인 서브시퀀스 매칭을 위하여 인덱스를 사용한다. 또한, 인덱싱의 단위로서 윈도우(window)라는 개념을 이용한다. 윈도우란 시퀀스로부터 추출되는 서브시퀀스로서, 사전에 결정된 고정된 길이 w의 크기를 갖는다. 서브시퀀스 매칭을 위하여 두 기법이 취하는 공통적인 아이디어는 다음과 같다.

먼저, 인덱스 구성을 위하여 각 시퀀스로부터 길이 w의 윈도우들을 추출하고, 각 윈도우를 이산 퓨리에 변환(discrete Fourier transform: DFT) 혹은 웨이블릿 변환(wavelet transform)을 이용하여 저차원 f-공간(f << w) 상의 윈도우 점(window point)으로 변환한다. 효과적인 서브시퀀스 매칭을 위하여 이러한 윈도우 점들을 다차원 인덱스(multidimensional index)의 하나인 R*-트리[Bec90]에 저장한다.

허용치가 ε인 서브시퀀스 매칭의 처리를 위하여 길이 l인 질의 시퀀스로부터 길이 w의 윈도우들을 추출하고, 각 윈도우를 DFT 혹은 웨이블릿 변환(wavelet transform)을 이용하여 저차원 f-공간(f << w) 상의 윈도우 점으로 변환한다. 각 윈도우 점에 대하여 $\varepsilon/\sqrt{p}$ ($p = \lfloor l/w \rfloor$)를 허용치로 갖는 범위 질의를 R*-트리 상에서 수행한다. 본 논문에서는 이 과정을 인덱스 검색 단계(index search step)라 부른다. 이러한 인덱스 검색 단계의 결과, 질의 시퀀스와 ε-매치 할 가능성이 높은 많은 후보 서브시퀀스(candidate subsequence)들이 반환된다. 그 다음, 착오 채택(false alarm)[Agr93][Fal94]을 해결하기 위하여 이 후보 서브시퀀스들을 포함하는 각 시퀀스를 디스크로부터 액세스하여 후보 서브시퀀스가 질의 시퀀스와 실제로 ε-매치 하는가의 여부를 판단한다. 본 논문에서는 이 과정을 후처리 단계(post-processing step)라 부른다.

FRM과 Dual-Match에서 윈도우의 크기를 어떻게 결정하는가에 따라 서브시퀀스 매칭의 성능이 많이 좌우된다. 즉, 질의 시퀀스의 길이와 윈도우 크기의 차이가 클수록 검색 성능이 저하된다. 이러한 현상을 윈도우 크기 효과(window size effect)라고 한다[Moo01]. FRM과 Dual-Match에서는 최소 질의 시퀀스의 길이를 기준으로 윈도우 크기를 설정하여 단 하나의 인덱스를 구성하고, 이를 이용하여 서브시퀀스 매칭을 수행한다. 그러나 이러한 방법은 질의 시퀀스의 길이가 증가할수록 서브시퀀스 매칭의 성능이 저하되는 문제점을 안고 있다. 이러한 문제를 해결하기 위하여 입력 가능한 모든 질의 시퀀스의 길이에 대하여 각각 인덱스를 생성하는 방법을 생각해 볼 수 있으나, 인덱스 관리 비용이 지나치게 증가하므로 실제 응용에 적용하는 데에는 현실성이 떨어지게 된다.

본 논문에서는 이러한 문제점을 해결하고자 인덱스 보간법(index interpolation) [Loh00]에 기반한 새로운 서브시퀀스 매칭 기법을 제안한다. 인덱스 보간법이란 둘 이상의 인덱스를 구축하고 주어진 질의 시퀀스의 길이에 따라 하나의 적절한 인덱스를 선택하여 서브시퀀스 매칭을 수행하는 기법이다.

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

2. 연구 동기

2.1. 윈도우 크기 효과

FRM 및 Dual-Match를 이용한 서브시퀀스 매칭에서는 R*-트리에 저장된 윈도우의 크기가 작을수록 착오 채택의 수가 증가하는 경향이 있다. 예를 들어, 두 R*-트리 R1과 R2에 대하여 R1이 R2보다 큰 윈도우를 이용하여 구성되었다고 가정하자. 이때, R2를 이용한 인덱스 검색 단계를 통하여 반환된 후보 집합 S2 내에는 R1을 이용한 인덱스 검색 단계를 통하여 반환된 후보 집합 S1에는 포함되지 않는 후보 서브시퀀스가 존재할 수 있다. 착오 채택이 증가는 후처리 시간의 증가를 초래하며, 전체 수행 시간이 길어지게 된다. 이러한 경향을 윈도우 크기 효과(window size effect)라 한다[Moo01]. 따라서 큰 윈도우를 사용하여 R*-트리를 구성하는 것은 전체 수행 시간을 개선시키는데 좋은 효과가 있다.

반면, 저장된 윈도우의 크기가 질의 시퀀스 크기보다 크다면, 해당 R*-트리를 사용할 수 없다[Fal94]. 따라서 이 경우에는 순차 검색(sequential scan)에 의하여 서브시퀀스 매칭을 수행해야 하므로 처리 시간이 매우 길어진다. 따라서 다차원 인덱스에 저장된 윈도우의 크기를 올바르게 선택함으로써 전체 서브시퀀스 매칭의 성능을 최적화 할 수 있다.

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

2.2. 사전 실험 결과 및 분석

그림 1은 사전 실험 결과를 보인 것이다. 서브 시퀀스 매칭을 위한 기법으로

는 FRM에 비하여 더 우수한 성능을 보이는 것으로 규명된 바 있는[Moo01] Dual-Match를 이용하였다.

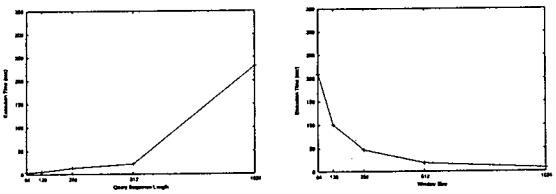

그림 1. 질의 시퀀스 길이와 윈도우 크기에 따른 실행 시간의 변화

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

3. 제안하는 기법

본 장에서는 서브시퀀스 매칭 수행 시에 윈도우 크기 효과에 의한 성능 저하를 극복하기 위한 방법으로 인덱스 보간법(index interpolation)[Loh00]을 이용하는 새로운 기법을 제안한다. 인덱스 보간법은 먼저 서로 다른 크기의 윈도우들을 대상으로 다수의 인덱스들을 구축한 후, 서브시퀀스 매칭시 주어진 질의 시퀀스의 길이에 가장 적합한 하나의 인덱스를 인덱스 검색 단계에서 사용하는 기법이다.

일반적으로, 구성된 인덱스들의 수가 늘어날수록 서브시퀀스 매칭의 처리 성능은 향상된다. 하지만, 늘어난 인덱스를 관리하기 위한 비용이 증가하는 문제가 발생된다. 인덱스 관리 비용은 인덱스를 저장하기 위한 물리적 공간뿐만 아니라, 데이터가 삽입, 삭제, 갱신될 때마다 해당 인덱스들을 변경하는 비용까지 포함한다. 따라서, 인덱스 보간법에서는 최적의 검색 성능을 지원할 수 있는 가능한 한 적은 수의 인덱스들을 선택하여 구성하는 것이 대단히 중요하다.

3.1. 윈도우 크기의 선택

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

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

$$w_{\max}(l_k) = max\{w_i \mid w_i \leq l_k \ (1 \leq i \leq m)\} \text{ FRM 또는}$$

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

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

보조 정리 1. 공식 (1)에 의해 선택된 윈도우 크기를 이용하여 수행한 인덱스 보간법 기반 서브시퀀스 매칭에서는 착오 기각이 발생하지 않는다.
증명: 참고문헌 [Lim05] 참조 □

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

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

3.2. 최적의 다중 인덱스 구성 방안

인덱스 보간법은 둘 이상의 인덱스를 사전에 구성해 두고, 주어진 질의 시퀀스에 대하여 가장 적합한 인덱스를 선택하여 서브시퀀스 매칭을 수행한다는 기본적인 원칙일 뿐 어떤 크기의 윈도우들을 대상으로 인덱스들을 구성할 것인가는 지정하지 않는다[Loh00]. 본 절에서는 물리적 데이터베이스 설계 기법을 이용한 최적의 서브시퀀스 매칭을 지원하는 다중 인덱스 구성 방안에 대하여 논의한다.

물리적 데이터베이스 설계는 주어진 질의 집합을 대상으로 최적의 질의 처리를 제공하는 물리적 파일 구조 및 액세스 구조를 결정하는 과정이다[Lee97]. 제안된 기법은 응용에서 사용될 질의 시퀀스들에 대한 정보로서 <길이, 빈도>의 쌍과 응용에서 구성할 인덱스의 개수 m을 입력으로 받아 서브시퀀스 매칭을 최적으로 처리할 수 있는 인덱스들을 구성할 수 있도록 윈도우 크기 $w_1, w_2, \cdots, w_m$를 결정한다. 먼저, 전체 서브시퀀스 매칭 처리를 위한 비용 공식을 도출하고, 그 공식에 기반하여 전체 질의 시퀀스들을 대상으로 최적의 서브시퀀스 매칭 수행을 위한 윈도우 크기를 선정하는 알고리즘을 제시한다.

주어진 질의 시퀀스의 길이 l_k와 응용에서의 사용 빈도수 f_k, 윈도우 크기 리스트 W를 이용하여 서브시퀀스 매칭을 수행하였을 때의 비용을 $C(l_k, f_k, W)$라고 표기한다. 본 연구에서는 이 비용을 아래와 같이 추정한다.

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

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

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

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

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

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

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

보조정리 2. 윈도우 크기 리스트 $W_{optimal} =< w_1, w_2, \ldots, w_m >$를 $L =< l_1, l_2, \ldots, l_n >$ 및 $F =< f_1, f_2, \ldots, f_n >$의 서브시퀀스들을 위한 최적의 윈도우 크기 리스트라 하면, $< w_1, w_2, \ldots, w_m > =< l_{g(1)}, l_{g(2)}, \ldots, l_{g(m)} >$이 성립한다. (여기서, $1 \leq g(1) < g(2) < \cdots g(m) \leq n$).
증명: 참고문헌 [Lim05] 참조 □

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

$C(L, F, W_{optimal})$를 구하는 과정은 크게 두 단계로 구성된다. 단계 1에서는 $n \times n$ 크기의 NC테이블 내의 각 요소 $NC(i,j)(1 \leq i \leq j \leq n)$에

$C(L[i..j], F[i..j], < l_i >)$값을 저장한다. $C(L[i..j], F[i..j], < l_i >)$는 윈도우 크기 리스트 $< l_i >$를 이용하여 $< l_i, ..., l_j >$의 부분리스트를 처리하는 비용이다. 단계 2에서는 NC테이블을 이용하여 최소 실행 비용인 $C(L, F, W_{optimal})$를 계산한다.

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

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

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

보조정리 4. $C'(i,j) = \min_{k=i+1}^{n-j+2} NC(i,k-1) + C'(k,j-1)$

증명: 참고문헌 [Lim05] 참조 □

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

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

4. 성능 평가

4.1. 실험 환경

본 연구에서는 성능 분석을 위하여 합성 데이터를 사용하였다. 합성 데이터 내의 각 시퀀스 $S = <s_1, s_2, ..., s_n>$ 는 다음과 같은 랜덤 워크(random walk) 형태를 가진다[Agr93].

$$s_i = s_{i-1} + z_i$$

여기서 z_i는 구간 [-0.1, 0.1] 사이에서 균일한 분포를 취하는 랜덤 변수이며, 시퀀스의 첫 요소 값 s_1은 구간 [1, 10] 사이의 임의의 값을 취하도록 하였다.

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

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

표 1. 그룹 내에 포함된 질의 시퀀스의 개수

다차원 인덱스로는 1KB 크기의 페이지를 갖는 R*-트리[Bec90]를 사용하였다. 인덱싱을 위한 저차원 변환은 DFT를 통하여 수행하였으며, 특성 추출 계수 f=6으로 하였다. 서브 시퀀스 매칭을 위한 기법으로는 사전실험에서와 마찬가지로 Dual-Match를 이용하였다.

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

4.2. 실험 결과

실험 1에서는 1024의 길이를 갖는 합성 데이터를 2000, 3000, 4000, 5000개로 개수를 증가시키면서, 세가지 기법의 성능의 변화를 관찰하였다. 기법(B)와 기법(C)에서는 각각 4개의 다중 인덱스를 사용하도록 설정하였다.

그림 2는 실험 1의 결과를 보인 것이다. 가로 축은 데이터 시퀀스의 개수, 세로 축은 실행 시간의 평균을 초 단위로 나타낸 것이다.

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

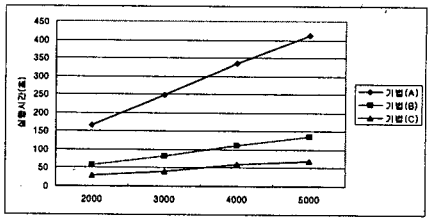

그림 2. 데이터 시퀀스 개수에 의한 성능 비교

실험 2에서는 2000, 3000, 4000, 5000의 길이를 갖는 1000개의 합성 데이터를 대상으로 세가지 기법의 성능의 변화를 관찰하였다. 실험 1에서와 마찬가지로, 기법(B)와 기법(C)에서는 각각 4개의 다중 인덱스를 사용하도록 설정하였다. 실험 결과는 실험 1과 그 경향이 매우 유사하게 나타났다. 지면 고나계상 결과 그래프는 생략한다. 기법(C)는 데이터 시퀀스의 길이에 상관 없이 기법(A)와 비교하여 약 6.2배, 기법(B)에 비하여 약 2배의 성능이 개선되는 것으로 나타났다.

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

5. 결론

본 논문에서는 기존의 기법에서 윈도우 크기 효과에 의하여 검색 성능이 저하되는 문제점을 해결하고자 인덱스 보간법 [Loh00]에 기반한 새로운 서브시퀀스 매칭 기법을 제안하였다. 인덱스 보간법이란 하나 이상의 인덱스를 구축하고, 주어진 질의 시퀀스의 길이에 따라 하나의 적절한 인덱스를 선택하여 검색을 수행함으로써 서브시퀀스 매칭 성능을 향상시키는 기법이다.

본 논문의 주요 공헌은 다음과 같다.

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

실험 결과, 제안된 기법은 단일 인덱스를 사용한 경우와 비교하여 데이터 시퀀스의 개수 및 길이에 상관 없이 약 6배의 성능 향상을 보였으며, 일정 간격의 인덱스를 사용한 경우와 비교하여 약 2배의 성능 향상을 보였다.

참고문헌

[Agr93] R. Agrawal, C. Faloutsos, and A. Swami, "Efficient Similarity Search in Sequence Databases," In *Proc. Int'l. Conf. on Foundations of Data Organization and Algorithms*, FODO, pp. 69-84, Oct. 1993.

[Bec90] N. Beckmann et al., "The R*-tree: An Efficient and Robust Access Method for Points and Rectangles," In *Proc. Int'l. Conf. on Management of Data*, ACM SIGMOD, pp. 322-331, May 1990.

[Cha99] K. P. Chan and A. W. C. Fu, "Efficient Time Series Matching by Wavelets," In *Proc. Int'l. Conf. on Data Engineering*, IEEE ICDE, pp. 126-133, 1999.

[Fal94] 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.

[Loh00] W. K. Loh, S. W. Kim, and K. Y. Whang, "Index Interpolation: An Approach for Subsequence Matching Supporting Normalization Transform in Time-Series Databases," In *Proc. ACM Int'l. Conf. on Information and Knowledge Management*, ACM CIKM, pp. 480-487, 2000.

[Moo01] Y. S. Moon, K. Y. Whang, and W. K. Loh, "Duality-Based Subsequence Matching in Time-Series Databases," In Proc. *Int'l Conf. on Data Engineering*, IEEE ICDE, pp. 263-272, 2001.

[Raf97] 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.

[Raf99] D. Rafiei, "On Similarity-Based Queries for Time Series Data," In *Proc. Int'l. Conf. on Data Engineering*, IEEE ICDE, pp. 410-417, 1999.

[Lim05] S. H. Lim, A Physical Database Design Approach for Index-Interpolation Based Time Series Subsequence-Matching, MS Thesis, Hanyang University, Korea, 2005.