정규화 변환을 지원하는 스트리밍 시계열 매칭 알고리즘

(An Efficient Algorithm for Streaming Time-Series Matching that Supports Normalization Transform)

노 웅 기 † 문 양 세 †† 김 영 국 †††
(Woong-Kee Loh) (Yang-Sae Moon) (Young-Kuk Kim)

요 약 최근에 센서 및 모바일 장비들의 발전으로 인하여 이러한 장비들로부터 생성된 대량의 데이타 스트림(data stream)의 처리가 중요한 연구 과제가 되고 있다. 데이타 스트림 중에서 연속되는 시점에 얻어진 실수 값들의 스트림을 **스트리밍 시계열**(*streaming time-series*)이라 한다. 스트리밍 시계열에 대한 유사성 매칭은 여러 가지 고유 특성에 의하여 기존의 시계열 데이타와는 다르게 처리되어야 한다. 본 논문에서는 정규화 변환(normalization transform)을 지원하는 스트리밍 시계열 매칭 문제를 해결하기 위한 효율적인 알고리즘을 제안한다. 기존에는 스트리밍 시계열을 아무런 변환 없이 비교하였으나, 본 논문에서는 정규화 변환된 스트리밍 시계열을 비교한다. 정규화 변환은 절대적인 값은 달라도 유사한 변동 경향을 가지는 시계열 데이타를 찾기 위하여 유용하다. 본 논문의 공헌은 다음과 같다. (1) 기존의 정규화 변환을 지원하는 서브시퀀스 매칭 알고리즘[4]에서 제시된 정리(theorem)를 이용하여 정규화 변환을 지원하는 스트리밍 시계열 매칭 문제를 풀기 위한 간단한 알고리즘을 제안한다. (2) 검색 성능을 향상시키기 위하여 간단한 알고리즘을 k ($\geq$ 1) 개의 인덱스를 이용하는 알고리즘으로 확장한다. (3) 주어진 k에 대하여, 확장된 알고리즘의 검색 성능을 최대화하기 위해 k 개의 인덱스를 생성할 최적의 윈도우 길이를 선택하기 위한 근사 방법(approximation)을 제시한다. (4) 스트리밍 시계열의 연속성(continuity) 개념[8]에 기반하여, 현재 시점 t_0에서의 스트리밍 서브시퀀스에 대한 검색과 동시에 미래 시점 ($t_0 + m - 1$) ($m \geq 1$)까지의 검색 결과를 한번의 인덱스 검색으로 구할 수 있도록 재차 확장한 알고리즘을 제안한다. (5) 일련의 실험을 통하여 본 논문에서 제안된 알고리즘들 간의 성능을 비교하고, k 및 m 값의 변화에 따라 제안된 알고리즘들의 검색 성능 변화를 보인다. 본 논문에서 제시한 정규화 변환 스트리밍 시계열 매칭 문제에 대한 연구는 이전에 수행된 적이 없으므로 순차 검색(sequential scan) 알고리즘과 성능을 비교한다. 실험 결과, 제안된 알고리즘은 순차 검색에 비하여 최대 13.2배까지 성능이 향상되었으며, 인덱스의 개수 k가 증가함에 따라 검색 성능도 함께 증가하였다.

키워드 : 정규화 변환, 스트리밍 시계열 매칭, 다중 인덱스, 근사 방법, 연속성

Abstract According to recent technical advances on sensors and mobile devices, processing of data streams generated by the devices is becoming an important research issue. The data stream of real values obtained at continuous time points is called *streaming time-series*. Due to the unique features of streaming time-series that are different from those of traditional time-series, similarity matching problem on the streaming time-series should be solved in a new way. In this paper, we propose an efficient algorithm for streaming time-series matching problem that supports normalization transform. While the existing algorithms compare streaming time-series without any transform, the algorithm proposed in the paper compares them after they are normalization-transformed. The normalization transform is useful for finding time-series that have similar fluctuation trends even though they consist of distant element values. The major contributions of this paper are as follows. (1) By using a theorem presented in the context of subsequence matching that supports normalization

· 본 논문은 2006년 정부(교육인적자원부)의 재원으로 한국학술진흥재단의 지원을 받아 수행한 연구임(KRF-2006-214-D00130)
† 정 회 원 : Univ. of Minnesota Dept. of CSE 연구원
lohw@cs.umn.edu
†† 정 회 원 : 강원대학교 컴퓨터학부 교수
ysmoon@kangwon.ac.kr
(Corresponding author임)
††† 정 회 원 : 충남대학교 전기정보통신공학부 교수
ykim@cnu.ac.kr
논문접수 : 2006년 8월 3일
심사완료 : 2006년 10월 30일

transform[4], we propose a simple algorithm for solving the problem. (2) For improving search performance, we extend the simple algorithm to use k (≥ 1) indexes. (3) For a given k, for achieving optimal search performance of the extended algorithm, we present an approximation method for choosing k window sizes to construct k indexes. (4) Based on the notion of *continuity*[8] on streaming time-series, we further extend our algorithm so that it can simultaneously obtain the search results for m (≥ 1) time points from present t_0 to a time point ($t_0 + m - 1$) in the near future by retrieving the index only once. (5) Through a series of experiments, we compare search performances of the algorithms proposed in this paper, and show their performance trends according to k and m values. To the best of our knowledge, since there has been no algorithm that solves the same problem presented in this paper, we compare search performances of our algorithms with the sequential scan algorithm. The experiment result showed that our algorithms outperformed the sequential scan algorithm by up to 13.2 times. The performances of our algorithms should be more improved, as k is increased.

Key words : Normalization Transform, Stream Time-series Matching, Multiple Indexes, Approximation, Continuity

1. 서 론

시계열 데이타(time-series data)는 연속되는 시점에 얻어진 일련의 실수(real number) 값들이며, 주가 데이타, 의료 기록 데이타, 상품 매출 데이타, 기후 관측 데이타 등의 예를 들 수 있다[1-3]. 대개의 시계열 데이타는 일정 간격의 시간마다 얻어지며, 전체 길이는 수십 년에 해당하기도 한다. 이러한 대량의 데이타를 효율적으로 처리하기 위하여 많은 연구들이 수행되어 왔다[1, 2,4-6]. 그러한 연구들 중에 유사 시퀀스 매칭(similar sequence matching)에 대한 연구는 데이타베이스에 저장된 대량의 시계열 데이타 중에서 사용자로부터 주어진 질의 시퀀스와 유사한 시계열 데이타를 찾기 위한 것으로, 데이타 마이닝 분야의 중요한 연구 과제 중의 하나이다. 이러한 기존의 시계열 데이타에 대한 검색은 시계열 데이타가 데이타베이스에 저장되어 있으며 갱신이 거의 일어나지 않는 정적인 환경을 가정하고 있다. 최근에는 센서 및 모바일 장비들의 발전으로 인하여 이러한 장비들로부터 생성된 데이타의 처리가 중요한 연구 과제가 되고 있다. 이러한 데이타를 **데이타 스트림** (*data stream*)이라 하며, 일반적으로 데이타 스트림은 연속된 관계형 레코드(relational record)의 형태를 가진다. 데이타 스트림 중에서 연속되는 시점에 얻어진 실수 값들의 스트림을 **스트리밍 시계열**(*streaming time-series*)이라 하고[7,8], 스트리밍 시계열에 대한 유사성 매칭에 대한 연구가 최근 활발히 수행되고 있다[7-9].

스트리밍 시계열은 다음과 같은 특성에 의하여 기존의 시계열 데이타와는 다르게 처리되어야 한다[10]. 첫째, 스트리밍 시계열의 요소 값들은 온라인으로 처리된다. 하나의 스트리밍 요소 값이 입력됨과 동시에 그 값에 대한 처리를 시작해야 하며, 대개 다음 요소 값이 입력되기 전에 그 처리를 마쳐야 한다. 반면, 시계열 데이타에 대한 처리는 시간적인 한계를 두고 있지 않다. 둘째, 스트리밍 시계열은 무한대의 길이를 가진다고 가정한다. 따라서, 스트리밍 시계열은 데이타베이스에 저장할 수 없으며, 데이타베이스에 전체 시계열 데이타가 저장되어 있다고 가정하고 있는 기존의 알고리즘은 그대로 적용할 수 없다. 셋째, 일정 시간 이전 과거의 스트리밍 시계열은 다시 볼 수 없거나 보기가 매우 어렵다. 스트리밍 시계열은 무한대의 길이를 가진다고 가정하므로, 과거의 데이타는 버리거나 압축 등의 방법을 통하여 저장한다. 따라서, 모든 시계열 데이타를 볼 수 있다고 가정하고 있는 기존의 알고리즘은 그대로 적용할 수 없다.

본 논문에서는 정규화 변환(normalization transform)을 지원하는 스트리밍 시계열 매칭 문제를 해결하기 위한 효율적인 알고리즘을 제안한다. 정규화 변환[4,11]은 절대적인 값은 달라도 유사한 변동 경향을 가지는 시계열 데이타를 찾기 위하여 유용하다[4,12]. 예를 들어, 유사한 가격 경향을 가지는 주식이나 유사한 판매 경향을 가지는 물품의 검색에 적용할 수 있다. 그림 1은 한국 증권시장에 상장된 통신 회사의 실제 주가 데이타를 보인 것이다. 그림에서 검은색으로 나타낸 그래프는 주가 데이타이고, 굵은 녹색으로 나타낸 그래프는 10-이동평균 그래프이다. 10-이동평균 그래프는 과거 10일 동안의 주가의 평균을 계산하여 연결한 그래프로, 잡음(noise)을 없애기 위하여 활용된다. 이러한 이동평균 그래프에 대하여 정규화 변환을 수행하여 유사한 추세를 갖는 시계열 데이타를 좀더 정확하게 검색할 수 있다[12]. 그림의 시점 A와 B에서 이동평균 그래프가 상향 반전하는 추세를 보이고 있는 반면, 시점 C에서는 하향 반전하는 추세를 보이고 있다. 만약 정규화 변환을 수행하지 않고 각 시점 주위의 이동평균 그래프를 비교하면, 시점 B와 C 주위의 그래프가 동일하게 180,000원 주변의 가격을 가지므로 시점 A보다 유사한 것으로 판정한

다. 하지만, 정규화 변환을 수행한 후에는 시점 A와 B 주위의 그래프가 유사한 (상향 반전하는) 추세를 가지므로 시점 C보다 유사한 것으로 판정한다. 이러한 주가 데이타는 주식 거래가 진행되는 동안에 계속하여 생성되므로, 스트리밍 시계열에 대한 검색 알고리즘을 필요로 한다. 실제의 펀드 매니저들은 분 또는 초 단위로 매매 판단을 내려야 하는 경우가 많으므로, 본 논문에서 제시하는 기법이 매우 유용할 것으로 판단된다.

기존 논문에서는 스트리밍 시계열을 아무런 변환 없이 비교하였으나, 본 논문에서는 정규화 변환된 스트리밍 시계열을 비교한다. 본 논문의 공헌은 다음과 같다. (1) 기존의 정규화 변환을 지원하는 서브시퀀스 매칭 알고리즘[4]에서 제시된 정리(theorem)를 이용하여 정규화 변환을 지원하는 스트리밍 시계열 매칭 문제를 풀기 위한 간단한 알고리즘을 제안한다. (2) 검색 성능을 향상시키기 위하여 간단한 알고리즘을 k (≥ 1)개의 인덱스를 이용하는 알고리즘으로 확장한다. (3) 주어진 k에 대하여, 확장된 알고리즘의 검색 성능을 최대화하기 위해 k 개의 인덱스를 생성할 최적의 윈도우 길이를 선택하기 위한 근사 방법(approximation)을 제시한다. (4) 스트리밍 시계열의 연속성(continuity) 개념[8]에 기반하여, 현재 시점 t_0에서의 스트리밍 서브시퀀스에 대한 검색과 동시에 미래 시점 ($t_0 + m - 1$) ($m \geq 1$)까지의 검색 결과를 한번의 인덱스 검색으로 구할 수 있도록 재차 확장한 알고리즘을 제안한다. (5) 일련의 실험을 통하여 본 논문에서 제안된 알고리즘들 간의 성능을 비교하고, k 및 m 값의 변화에 따라 제안된 알고리즘들의 검색 성능 변화를 보인다. 본 논문에서 제시한 정규화 변환 스트리밍 시계열 매칭 문제에 대한 연구는 이전에 수행된 적이 없으므로 순차 검색(sequential scan) 알고리즘과 성능을 비교한다. 실험 결과, 제안된 알고리즘은 순차 검색에 비하여 최대 13.2배까지 성능이 향상되었으며, 인덱스의 개수 k가 증가함에 따라 검색 성능도 함께 증가하였다.

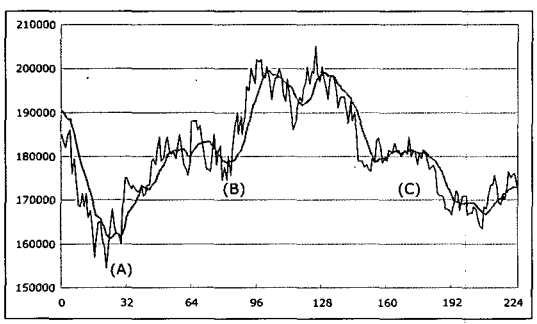


그림 1 실제 주가 및 이동평균 데이타

본 논문의 구성은 다음과 같다. 먼저, 제2절에서는 스트리밍 시계열에 대한 기존의 유사성 매칭 연구들에 대하여 간략하게 설명한다. 제3절에서는 본 논문에서 해결할 정규화 변환 스트리밍 시계열 매칭 문제에 대하여 공식적으로 정의한다. 제4절에서는 문제 해결을 위한 간단한 알고리즘과 k 개의 인덱스를 사용하는 확장된 알고리즘에 대하여 설명한다. 제5절에서는 확장된 알고리즘의 검색 성능을 최대화하기 위하여 최적의 윈도우의 길이를 선택하는 방법을 설명한다. 제6절에서는 연속성 개념에 기반하여 검색 성능을 좀더 개선하기 위해 재차 확장된 알고리즘에 대하여 설명한다. 제7절에서는 제안된 알고리즘들의 성능을 평가한다. 마지막으로, 제8절에서는 본 논문을 요약한다.

2. 관련 연구

본 절에서는 참고문헌 [7-9]에서 수행된 스트리밍 시계열에 대한 유사성 매칭 연구들에 대하여 설명한다. 기존의 시계열 데이타에 대한 유사성 매칭 연구들에 대해서는 참고문헌 [1,2,4-6]을 참고하기 바란다.

참고문헌 [7]에서는 예측(prediction)에 기반한 유사성 매칭 알고리즘을 제안하였다. 이 알고리즘은 현재까지 입력된 스트리밍 시계열을 기반으로 앞으로 입력될 스트리밍 시계열을 예측하여, 질의 시퀀스와 예측된 스트리밍 시계열 간의 거리를 미리 계산한다. 이때, 하나의 질의 시퀀스와 예측된 스트리밍 시계열 내의 여러 시점의 스트리밍 서브시퀀스와의 거리를 동시에 계산하기 위하여 참고문헌 [7]에서는 고속 푸리에 변환(Fast Fourier Transform, FFT)을 사용하였다. 예측된 값들에 대한 실제 값들이 도착했을 때에 두 값들 간의 오차를 보정하여 정확한 질의 시퀀스만을 반환한다. 이 알고리즘은 미래의 스트리밍 시계열에 대한 검색을 미리 수행할 수 있다는 장점이 있는 반면, 데이타베이스 내의 모든 질의 시퀀스 각각에 대하여 거리 계산을 수행하여야 하며, 먼 미래에 대하여 예측 오차가 커지면 오차 보정을 위한 부담(overhead)이 증가한다는 단점이 있다.

참고문헌 [8]에서는 선인출(prefetching)에 기반한 k-최근접 객체(k-nearest neighbor, k-NN) 검색 알고리즘을 제안하였다. 이 알고리즘은 가장 최근의 스트리밍 서브시퀀스로부터의 거리가 가장 가까운 k 개의 질의 시퀀스를 검색한다. 참고문헌 [7]의 알고리즘은 질의 시퀀스가 모두 메인 메모리에 적재(load)되어 있음을 가정하는 반면, 참고문헌 [8]에서는 모든 질의 시퀀스가 디스크에 저장되어 있음을 가정한다. 참고문헌 [8]의 알고리즘은 디스크에 저장된 질의 시퀀스를 선인출하여 검색 성능을 높이기 위하여, 시간적으로 근접한 스트리밍 서브시퀀스에 대한 검색 결과가 유사하다는 연속성 개

넘에 기반하여 검색을 수행한다. 이 알고리즘은 인덱스를 이용하여 대량의 질의 시퀀스에 대한 검색을 수행한다는 장점이 있는 반면, 고정된 길이의 질의 시퀀스만을 지원한다는 단점이 있다.

참고문헌 [9]에서는 주식 가격과 같은 금융 데이타 스트림(financial data stream)에 대한 유사 서브시퀀스 매칭 알고리즘을 제안하였다. 이 알고리즘은 금융 데이타 스트림을 먼저 세분화(segmentation)하여 일련의 세그멘트(segment)로 변환하고 임의의 서브시퀀스를 하나의 세그멘트 리스트로 표현한다. 기존의 참고문헌 [7,8]에서는 스트리밍 시계열을 구성하는 요소 값들을 직접 이용하여 유사성 검색을 수행한 반면, 이 알고리즘은 세그멘트를 이용하여 유사성 검색을 수행한다. 이 알고리즘은 엘리엇 파동 이론(Elliott Wave Theory)[13]에 기반하여 유사성에 대한 새로운 정의를 내렸고, 새로운 데이타 스트림 요소 값이 입력될 때마다 데이타 스트림이 검색하고자 하는 등락 경향과 유사한 경향을 갖는지 검토한다. 참고문헌 [9]에서 다루는 문제는 데이타 스트림의 경향에 따른 검색이라는 점에서 본 논문에서 다루고자 하는 문제와 비슷하지만, 특정 응용 분야에 특화된 문제라는 점과 좀더 세밀하고 다양한 형태의 검색을 수행하기 어렵다는 단점을 갖는다.

3. 문제 정의

본 논문에서는 정규화 변환을 지원하는 스트리밍 시계열 매칭 문제를 다룬다. 정규화 변환은 다음과 같이 정의된다[4,11].

정의 1. 길이 n인 시퀀스 $X = (x_0, ..., x_{n-1})$에 대하여 정규화 변환한 시퀀스를 $\nu(X)$라 표기하고, $\nu(X)$의 각 요소 값 $\tilde{x}_i\,(0 \le i < n)$는 다음과 같이 얻어진다:

$$\tilde{x}_i = \frac{x_i - \mu(X)}{\sigma(X)} \quad . \tag{1}$$

여기에서, $\mu(X)$와 $\sigma(X)$는 각각 시퀀스 X를 구성하는 모든 요소 값들의 평균(mean)과 표준편차(standard deviation)이다. □

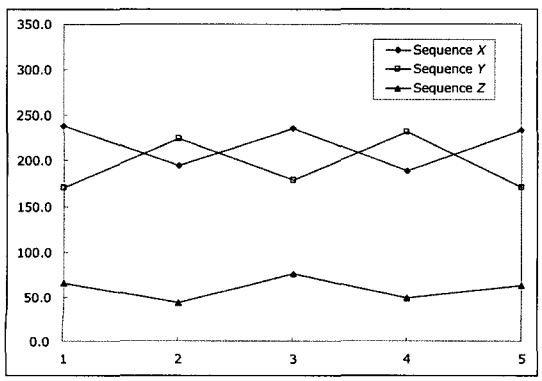
표준편차 $\sigma(X) = 0$인 경우는 시퀀스 X를 구성하는 모든 요소 값들이 동일한 경우이며, 이러한 경우는 별도로 처리한다. 그림 2는 정규화 변환의 예를 보이고 있다. 그림 2(a)는 정규화 변환 전, 그림 2(b)는 정규화 변환 후의 세 시퀀스 X, Y, Z를 보이고 있다. 그림에서 보는 바와 같이, 정규화 변환 전에는 시퀀스 X와 Y가 유사한 것으로 보이나, 정규화 변환 후에는 $\nu(X)$와 $\nu(Z)$가 유사한 변동 경향을 가지는 것으로 보인다.

정규화 변환을 지원하는 스트리밍 시계열 매칭 문제의 입력으로는 무한한 길이의 하나의 스트리밍 시계열 S와 임의의 길이를 가지는 N 개의 질의 시퀀스 Q_i $(0 \le i < N)$, 검색 한계(threshold) ε이 주어진다. 질의 시퀀스의 길이 분포는 특정 분포를 가정하지 않으며, 동일한 길이의 질의 시퀀스가 임의의 개수 존재할 수 있다. 출력으로는 스트리밍 시계열 S 내의 가장 최근의 서브시퀀스 X와 유사한 변동 경향을 가지는 질의 시퀀스 Q_i를 반환한다. 스트리밍 시계열 S 내의 가장 최근의 서브시퀀스 X를 **스트리밍 서브시퀀스**(*streaming subsequence*)라 부른다. 유사성 조건을 공식으로 표현하면 아래와 같다:

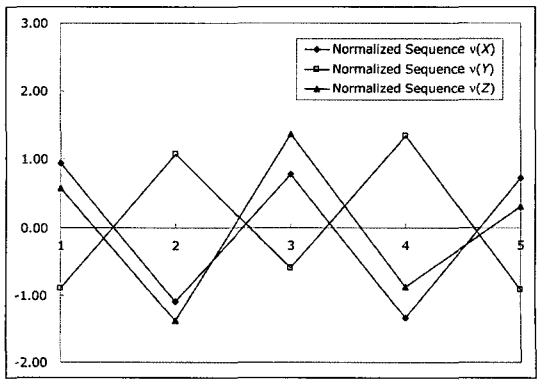
$$d(\nu(X), \nu(Q_i)) \le \varepsilon \quad . \tag{2}$$

여기에서, 함수 $d()$는 임의의 두 시퀀스 간의 거리를 반환하며, 본 논문에서는 기존 논문에서 가장 많이 적용된 유클리드 거리(Euclidean distance)를 사용한다. $\nu(X)$ 및 $\nu(Q_i)$는 각각 X 및 Q_i를 정규화 변환한 결과를 의미한다.

그림 3은 정규화 변환 스트리밍 서브시퀀스 매칭 과정을 보인 것이다. 그림에서 질의 시퀀스 Q_i $(0 \le i <$



(a) 정규화 변환 전의 시퀀스



(b) 정규화 변환 후의 시퀀스

그림 2 정규화 변환의 예

N)와 동일한 길이의 스트리밍 서브시퀀스 X를 추출하여 정규화 변환한 다음 유사성을 비교한다. 만약 정규화 변환된 질의 시퀀스 Q_i와 스트리밍 서브시퀀스 X 간의 거리가 ε보다 작으면 질의 시퀀스 Q_i를 반환한다. 표 1은 본 논문에서 사용하는 일반적인 표기법들을 정리한 것이다.

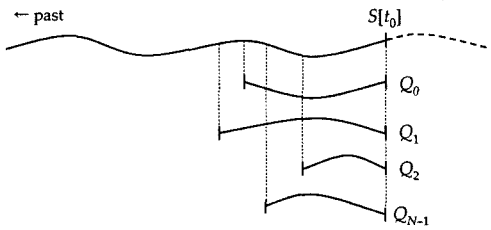


그림 3 정규화 변환 스트리밍 서브시퀀스 매칭

4. 제안된 알고리즘

본 논문에서는 기존의 참고문헌 [4]에서 제시된 정리를 이용하여 문제를 해결한다. 참고문헌 [4]에서는 임의의 두 시퀀스 X와 Q를 정규화 변환한 두 시퀀스 $v(X)$와 $v(Q)$에 대하여 다음의 정리 1과 같은 관계를 증명하였다.

정리 1. 동일한 길이의 임의의 두 시퀀스 X와 Q에 대하여 다음 공식이 성립한다:

$$d(v(X),v(Q)) \le \varepsilon \Rightarrow d(v(X_w),v(Q_w)) \le \varepsilon' . \quad (3)$$

여기에서, $X_w=(x_s,\ldots,x_f)$ 및 $Q_w=(q_s,\ldots,q_f)$ ($w = f - s + 1$)는 길이 w이며 각각 시퀀스 X 및 Q 내의 같은 오프셋 위치(s)에 포함되는 윈도우이다. 새로운 검색 한계 ε'은 다음과 같이 구해진다:

$$\varepsilon' = \sqrt{2w - 2\sqrt{w^2 - w\cdot\varepsilon^2 \cdot \frac{\sigma^2(X)}{\sigma^2(X_w)}}} . \quad (4)$$

여기에서, $\sigma(X)$와 $\sigma(X_w)$는 각각 시퀀스 X와 윈도우 X_w를 구성하는 값들의 표준편차이다.

증명. 참고문헌 [4] 참조. □

만약 공식 (4)를 이용하여 ε'을 계산하는 과정에서 안쪽 근호(root) 내의 값이 0보다 작은 경우가 발생하면, 공식 (4)를 사용할 수 없으며 순차 검색을 수행하여야 한다. 본 논문에서의 실험 결과, 질의 시퀀스의 길이가 충분히 길면 그러한 경우는 거의 발생하지 않았다. 공식 (4)의 바깥쪽 근호 내의 값은 항상 0보다 크다. 정리 1은 정규화 변환된 시퀀스 X와 질의 시퀀스 Q 간의 유사성 문제를 정규화 변환된 윈도우 X_w와 Q_w 간의 유사성 문제로 전환한다. 시퀀스 X와 질의 시퀀스 Q는 임의의 길이를 가질 수 있으므로, 모든 가능한 길이에 대하여 인덱스를 생성하기 어렵다. 반면에, 윈도우 X_w와 Q_w의 길이는 고정되어 있으므로, 고정 윈도우 길이에 대한 인덱스만을 생성하여 검색에 사용할 수 있다. 따라서, 공간 사용의 효율성 및 검색 성능의 향상을 거둘 수 있다.

본 논문에서는 정리 1을 이용하여 정규화 변환 스트리밍 서브시퀀스 매칭 문제를 해결한다. 먼저 간단한 알고리즘에 대해 설명하고, 성능 향상을 위하여 k (≥ 1) 개의 인덱스를 이용한 확장된 알고리즘에 대하여 설명한다. 제5절에서는 제안된 알고리즘의 성능을 최대화하기 위한 k 개의 윈도우 길이를 선택하는 방법을 설명하고, 제6절에서는 스트리밍 시계열의 연속성을 이용하여 성능을 더욱 향상시키도록 알고리즘을 재차 확장한다.

4.1 간단한 알고리즘

간단한 알고리즘은 모든 질의 시퀀스와 스트리밍 시계열에 대하여 동일한 길이의 최우측의 윈도우를 추출하여 인덱싱하고 검색을 수행하는 방법이다. 즉, 모든 질의 시퀀스에 대하여 길이 w_s인 최우측의 윈도우를 추출하여 인덱스를 생성하고, 스트리밍 시계열에 새로운

표 1 표기법 정리

표기법	설명
S	무한 길이의 스트리밍 시계열
$S[t]$	스트리밍 시계열 S 내의 시점 t의 요소 값. 현재 시점은 t_0, 스트리밍 시계열 S 내의 가장 최근 값은 $S[t_0]$로 표기함
X	스트리밍 시계열 S 내의 가장 최근의 스트리밍 서브시퀀스. $X=(S[t_0-Len(X)+1],\ldots,S[t_0])$
$X^{(t)}$	시점 t의 스트리밍 서브시퀀스. 현재 시점의 스트리밍 서브시퀀스는 $X=X^{(t_0)}$로 표기함
Q_i	질의 시퀀스 ($0 \le i < N$)
$T[s,f]$	임의의 시퀀스 T 내의 s-번째부터 f-번째 값으로 구성된 윈도우. $T[s,f]=(t_s,\ldots,t_f)$
$Len(T)$	임의의 시퀀스 T의 길이
N	질의 시퀀스의 개수
n	질의 시퀀스의 개수 (동일한 길이의 질의 시퀀스는 하나로 셈)

요소 값이 추가될 때마다 마찬가지로 길이 w_s인 최우측의 윈도우를 추출하여 검색을 수행한다. 여기에서, 길이 w_s는 최단 질의 시퀀스 길이이고, 질의 시퀀스와 스트리밍 시계열에서 추출한 윈도우를 각각 **질의 윈도우**(*query window*), **스트리밍 윈도우**(*streaming window*)라 부른다. 질의 윈도우는 인덱스에 저장하기 전에, 스트리밍 윈도우는 검색을 수행하기 전에 정규화 변환한다. 그림 4는 간단한 알고리즘의 매칭 과정을 보인 것이다. 최단 길이의 질의 시퀀스 Q_2의 길이 $Len(Q_2)$를 갖는 질의 윈도우 및 스트리밍 윈도우를 추출하여 유사성을 비교한다.

그림 5는 간단한 알고리즘의 인덱싱 단계와 검색 단계를 정리한 것이다. 인덱싱 단계의 라인 (2)에서는 질의 시퀀스 Q_i로부터 질의 윈도우 $Q_i[Len(Q_i) - w_s, Len(Q_i) - 1]$을 추출한다. 라인 (4)의 다차원 인덱스는 R^*-트리[14]를 비롯한 임의의 다차원 인덱스를 사용할 수 있다. 일반적으로, 데이타의 차원이 증가할수록 다차원 인덱스를 이용한 검색 비용이 기하급수적으로 증가하므로, 데이타의 차원을 감소시켜 다차원 인덱스에 저장한다[15]. 유사 시계열 매칭에서와 마찬가지로, 라인 (4)에서도 정규화 변환된 윈도우를 DFT 등을 통하여 먼저 f ($<w_s$)-차원 변환한 다음에 f-차원 인덱스에 저장한다. 검색 단계 알고리즘은 현재 시점 t_0의 스트리밍 서브시퀀스에 대한 검색을 수행한다. 검색 단계의 라인 (1)에서 스트리밍 시계열 S 내의 길의 w_s인 스트리밍 윈도우 $X_w = (S[t_0 - w_s + 1], \ldots, S[t_0])$를 추출한다. 라인 (3)에서 스트리밍 윈도우 $\nu(X_w)$에 대해서도 인덱싱 단계에서와 같은 방법으로 f-차원 변환한 다음에 검색을 수행한다.

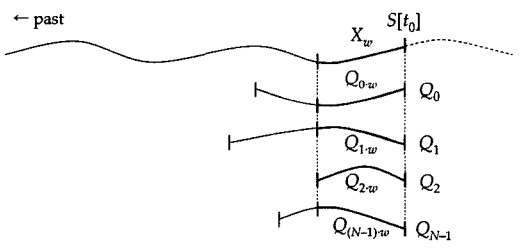


그림 4 간단한 알고리즘

그림 5(b)의 검색 단계의 라인 (3)에서 공식 (4)를 이용하여 ε'을 구할 때, 스트리밍 서브시퀀스 X에 대한 $\sigma(X)$ 값이 필요하다. 여기에서 구해지는 ε'은 모든 질의 시퀀스에 대하여 사용하는 한계이므로, $\sigma(X)$ 값을 구함에 있어서도 모든 질의 시퀀스와 같은 길이의 스트리밍 서브시퀀스 X에 대하여 구해야 한다. 즉, 아래와 같은 공식을 통하여 구할 수 있다:

$$\sigma_{\max}(X) = \max\{\sigma(X) \mid w_s \le Len(X) \le L\}. \quad (5)$$

여기에서, L은 최대 질의 시퀀스 길이를 의미한다. $\sigma_{\max}(X)$를 이용하여 ε'을 구하면 ε' 값이 최대가 되므로, 인덱스 검색에서 누락되는 질의 윈도우는 없다. 일

(1) **for** each query sequence Q_i ($0 \le i < N$) **do**
(2) Extract rightmost window Q_w of length w_s from Q_i;
(3) Normalization-transform Q_w;
(4) Store $\nu(Q_w)$ in a multidimensional index;
(5) **end for**.

(a) 인덱싱 단계

(1) Extract rightmost window X_w of length w_s from streaming time-series S;
(2) Normalization-transform X_w;
(3) Find the query windows $\nu(Q_w)$ within distance ε' from $\nu(X_w)$
using the index constructed in the index phase;
(4) **for** each window Q_w found **do**
(5) Identify the query sequence Q_i containing the window Q_w;
(6) Compute actual distance between $\nu(Q_i)$ and $\nu(X)$,
where X is streaming subsequence of the same length as Q_i in S;
(7) Return Q_i if the distance is not greater than ε;
(8) **end for**.

(b) 검색 단계

그림 5 간단한 알고리즘의 인덱싱 및 검색 단계

반적으로, ε'이 커지면 착오 채택(*false alarm or* false positive)이 커져 성능 감소의 한 원인이 될 수 있다. 하지만, 공식 (4)에서 $\sigma^2(X)$는 $\sigma^2(X_w)$로 나누어지고 두 개의 근호 내에 포함되어 있으므로 $\sigma(X)$가 커지더라도 ε'은 크게 증가하지 않는다.

제안된 간단한 알고리즘은 착오 기각(false dismissal *or* false negative)이 발생하지 않는다. 제안된 알고리즘은 인덱스를 통하여 정리 1의 공식 (3)의 결론부(consequent part)를 만족하는 모든 질의 윈도우 Q_w를 찾는다. 이러한 Q_w를 포함하는 모든 질의 시퀀스 Q_i의 집합은 공식 (3)의 조건부(antecedent part)를 만족하는 모든 질의 시퀀스 Q_i의 집합을 포함하는 집합이므로 착오 기각이 발생하지 않는다.

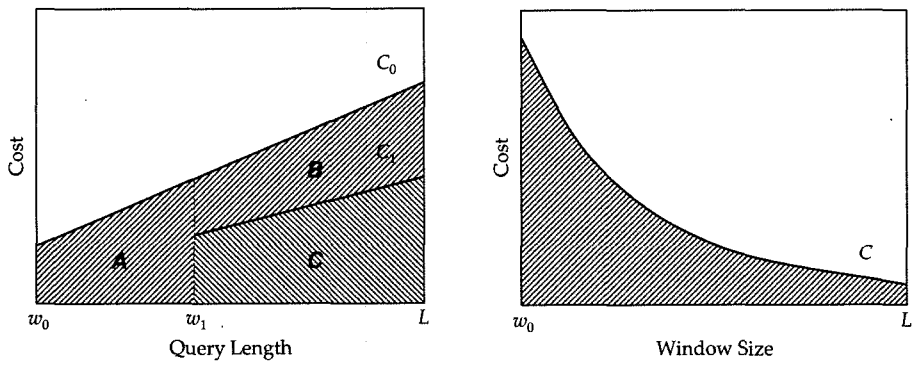
본 논문에서는 질의 시퀀스에서 추출한 윈도우들을 디스크 기반 인덱스를 사용하여 관리한다고 가정한다. 만일, 매칭 대상이 되는 질의 시퀀스의 개수가 매우 적은 경우라면 디스크 기반 인덱스 대신 메인 메모리 인덱스를 사용하는 것이 성능 면에서 훨씬 유리하다. 본 논문에서 제안된 알고리즘은 이러한 메인 메모리 인덱스를 사용하더라도 아무런 변화 없이 적용이 가능하다. 그러나, 응용 환경에 따라 전체 질의 시퀀스가 메인 메모리에 적재될 수 없는 경우도 많을 수 있다. 또한, 장착된 메모리 용량이 전체 질의 시퀀스 용량보다 크더라도, 매칭을 수행하는 많은 응용이 동시에 수행되는 환경이라면 전체 질의 시퀀스를 메모리에 적재하기 어렵게 된다. 따라서, 본 논문에서는 일반적인 디스크 기반 인덱스를 가정하고 설명을 전개한다.

4.2 간단한 알고리즘의 확장

앞에서 설명한 간단한 알고리즘은 다음과 같은 문제점이 있다. 간단한 알고리즘에서 인덱스를 생성하고 검색을 수행하기 위한 윈도우의 길이 w_s는 최단 질의 시퀀스의 길이이므로 다른 질의 시퀀스의 길이와 큰 차이가 있을 수 있다. 일반적으로, 정리 1을 이용하여 정규화 변환 스트리밍 서브시퀀스 매칭을 수행할 때, 질의 시퀀스와 윈도우 간의 길이 차이에 비례하여 검색 비용이 증가하고, 그 차이가 커지면 검색 성능이 크게 떨어질 수 있다.

그림 6은 질의 시퀀스 및 윈도우의 길이에 따른 검색 비용의 경향을 보인 것이다. 그림 6(a)는 고정 길이의 윈도우에 대하여 생성한 인덱스를 이용할 때에 질의 시퀀스의 길이에 따른 검색 비용의 경향을 보인 것이고, 그림 6(b)는 길이 L인 질의 시퀀스를 검색하기 위하여 사용되는 인덱스의 윈도우 길이에 따른 검색 비용의 경향을 보인 것이다. 이때, 질의 시퀀스의 길이는 균일 분포(uniform distribution)를 따른다고 가정한다. 그림 6(a)에서 비용 그래프 C_0가 윈도우 길이 $w_0(= w_s)$에 대하여 생성한 인덱스를 이용할 때의 검색 비용을 나타내며, L은 최대 질의 시퀀스 길이를 의미한다. 본 논문에서는 제7절에서 실험을 통하여 질의 시퀀스 및 윈도우의 길이에 따라 검색 비용이 그림 6과 같은 경향을 가짐을 확인한다.

확장된 알고리즘에서는 하나만의 인덱스를 사용하는 대신 k (≥ 1) 개의 인덱스를 사용하여 검색 성능을 높인다. 전체 질의 시퀀스를 k 개의 그룹으로 분할(partition)하여 각 그룹에 대하여 하나씩의 인덱스를 생성한다. 정규화 변환 스트리밍 서브시퀀스 매칭은 각 그룹에 대하여 생성된 인덱스를 이용하여 독립적으로 수행된다. 질의 시퀀스는 그 길이에 따라 k 개의 그룹으로 분할한다. 질의 시퀀스를 분할하기 위하여 선택된 k 개의 질의 시퀀스 길이를 w_0, ..., w_{k-1} ($w_0 < w_1 < ... < w_{k-1} < L$)이라 하고, j ($0 \leq j < k$)-번째 그룹을 G_j라 표기한다. 질의 시퀀스 Q는 다음의 공식에 따라 그룹 G_j에 속하게 된다:



(a) 질의 시퀀스의 길이에 따른 비용 (b) 윈도우 길이에 따른 비용

그림 6 질의 시퀀스 및 윈도우 길이에 따른 검색 비용

$$G_j = \{Q \mid w_j \le Len(Q) < w_{j+1}\} \quad (6)$$

이때, $w_k = L + 1$이라고 가정한다. 공식 (6)에 따라, 그룹 G_j에 포함된 질의 시퀀스의 최소 길이는 w_j이다. 그림 7은 $k = 2$ 개의 인덱스를 사용하도록 확장한 알고리즘의 검색 과정을 보인 것이다. 그림에서 질의 시퀀스 Q_0와 Q_1은 길이 w_1인 윈도우로 생성한 인덱스를 이용하여 검색하고, 질의 시퀀스 Q_2와 Q_N은 길이 w_0인 윈도우로 생성한 인덱스를 이용하여 검색한다.

그림 8은 확장된 알고리즘의 인덱싱 단계와 검색 단계를 정리한 것이다. 인덱싱 단계의 라인 (1)에서는 먼저 k 개의 질의 시퀀스의 길이를 선택한다. 최적의 검색 성능을 얻기 위하여 k 개의 질의 시퀀스 길이를 선택하는 방법에 대해서는 제5절에서 설명한다. 라인 (3)에서 각 그룹 G_j ($0 \le j < k$)에 대하여 간단한 알고리즘의 인덱싱 단계에서의 작업을 수행한다. 이때, 윈도우의 길이를 w_j로 하여 인덱스를 생성하며, 이 인덱스를 I_j라 표기한다. 검색 단계 알고리즘은 그림 5에서와 마찬가지로 현재 시점 t_0의 스트리밍 서브시퀀스에 대한 검색을 수행한다. 검색 단계의 라인 (2)에서 각 그룹 G_j에 대하여 스트리밍 시계열 S로부터 길이 w_j인 스트리밍 윈도우를 추출하고, 라인 (3)에서 인덱스 I_j를 이용하여 간단한 알고리즘의 검색 단계에서와 같이 유사한 질의 윈도우를 찾는다. 확장된 알고리즘에서는 각 그룹 G_j에 대하여 간단한 알고리즘과 동일한 검색을 수행하므로 착오 기각이 발생하지 않는다.

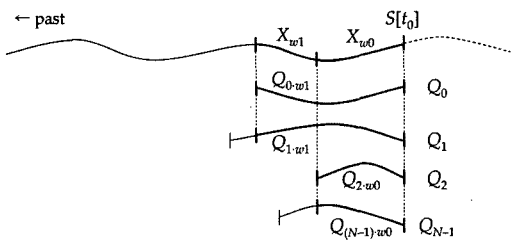


그림 7 확장된 알고리즘

확장된 알고리즘은 k 개의 인덱스를 생성한다는 점에서 참고문헌 [4]에서 제안한 인덱스 보간법(index interpolation)과 유사하다. 하지만, 본 논문에서 제안한 알고리즘은 두 가지 면에서 인덱스 보간법과 두 가지 중요한 차이점이 있다. 첫째, 인덱스 보간법에서는 동일한 윈도우가 k 개의 인덱스에 중복되어 저장되는 반면에, 제안된 알고리즘에서 하나의 윈도우는 k 개 중의 하나의 인덱스에만 저장된다. 따라서, 각 인덱스의 크기는 인덱스 보간법에 비하여 작아지고 검색의 효율성을 기할 수 있다. 둘째, 인덱스 보간법에서는 k 개의 인덱스 중에서 하나의 인덱스만을 선택하여 검색을 수행하였으나, 제안된 방법은 k 개의 인덱스를 모두 사용하여 검색을 수행한다. 각각의 인덱스를 이용하여 수행하는 검색은 서로 독립적이므로 병렬처리(parallel processing)가 가능하다. 확장된 알고리즘에서 k 값의 크기에 따른 절충 관계(trade-off)가 존재한다. 만약 k 값이 커지면, 각 질의 시퀀스 그룹 G_j에 대하여 수행되는 검색으로 인하여 발생하는 착오 채택이 감소하여 성능이 개선될 수 있으나, 검색해야 할 인덱스의 개수가 늘어나서 동시에 성능이 감소할 수 있다. 극단적으로 k가 N에 가까워지면 제안된 알고리즘은 순차 검색을 수행하는 것과 같아진다.

본 절에서 설명한 확장된 알고리즘은 기본적으로 정규화 변환을 수행함을 가정하고 있다. 즉, 전체 질의 시퀀스를 k 개의 그룹 G_j ($0 \le j < k$)로 분할하고 각 그룹에 대해 주어진 검색 범위인 ε 대신에 공식 (4)에 의한 새로운 검색 범위 ε'을 구한 후 검색을 실행할 수 있는 근거는 공식 (4)에서 w_j의 크기와 무관하게 ε'을

(1) Choose k window sizes w_0, ..., w_{k-1};
(2) Divide query sequences into k groups G_j ($0 \le j < k$) using Eq. (6);
(3) For each group G_j, do the same as in the indexing phase
in Simple Algorithm (Fig. 5(a));

(a) 인덱싱 단계

(1) **for** each group G_j ($0 \le j < k$) **do**
(2) Extract rightmost window X_w of length w_j from streaming time-series S;
(3) Do the same as in the searching phase in Simple Algorithm (Fig. 5(b))
using the index I_j constructed for the group;
(4) **end for**.

(b) 검색 단계

그림 8 확장된 알고리즘의 인덱싱 및 검색 단계

구할 수 있기 때문이다. 이러한 가정은 정규화 변환을 수행하지 않는 다른 시계열 매칭 알고리즘에는 일반적으로 적용 가능하지 않다.

본 논문에서 제안된 알고리즘은 각 질의 시퀀스 Q_i ($1 \le i < N$)에 대하여 서로 다른 검색 한계 ε_i를 설정하는 좀더 일반적인 문제 정의에도 쉽게 응용될 수 있다. 모든 질의 시퀀스에 대하여 동일한 검색 한계를 설정하는 경우, 만약 작은 검색 한계를 설정하면 길이가 긴 질의 시퀀스들이 짧은 길이의 질의 시퀀스에 비해 상대적으로 배제될 가능성이 높아지고, 만약 큰 검색 한계를 설정하면 의미 없는 짧은 질의 시퀀스들이 많이 반환될 가능성이 높아진다. 본 논문에서 제안된 알고리즘은 각 질의 시퀀스가 서로 다른 검색 한계를 가지는 일반적인 문제 정의에 대하여 다음과 같이 해결한다. 먼저, 검색 한계의 최대치, 즉 $\max\{\varepsilon_i\}$를 대표 검색 한계 ε으로 설정하여 제안된 알고리즘으로 검색을 수행한다. 이렇게 검색 한계를 설정하면 착오 기각이 발생하지 않음을 쉽게 증명할 수 있다. 후처리(post-processing) 과정에서, 착오 채택을 제거하기 위하여 제안된 알고리즘에 의해 반환된 질의 시퀀스 Q_i 각각에 대하여 대응되는 검색 한계 ε_i 내에 포함되는지 검사하여 조건을 만족하는 질의 시퀀스만을 최종적으로 식별한다. 본 절에서 설명한 확장된 알고리즘은 전체 질의 시퀀스를 그 길이에 따라 k 개의 그룹으로 분리하여 검색을 수행하며 각 그룹에 대하여 대표 검색 한계를 개별적으로 설정할 수 있으므로, 지나치게 큰 대표 검색 한계에 의한 착오 채택을 크게 줄일 수 있다. 본 논문에서는, 문제 정의 및 알고리즘 설명을 간단히 하기 위하여 모든 질의 시퀀스에 대해서 동일한 검색 한계가 주어진다고 가정한다.

5. 윈도우 길이 선택

본 절에서는 주어진 k에 대하여 검색 성능을 최대화할 수 있도록 윈도우의 길이 w_0, ..., w_{k-1}을 선택하는 방법에 대하여 설명한다. 앞 절에서 설명한 바와 같이, k 개의 질의 시퀀스 길이를 선택하고, 그 길이에 따라 질의 시퀀스를 k 개의 그룹으로 분할한다. 각 그룹 G_j에 대하여 선택된 질의 시퀀스 길이를 윈도우 길이로 하여 하나의 인덱스 I_j를 생성한다. 이때, 윈도우의 길이는 항상 해당 그룹 내의 모든 질의 시퀀스의 길이보다 작아야 하므로, w_0는 최소 질의 시퀀스 길이가 된다. 따라서, 본 절에서는 (k - 1) 개의 질의 시퀀스의 길이를 선택하는 방법을 설명한다.

검색 성능을 최대화하도록 (k - 1) 개의 질의 시퀀스의 길이를 선택하는 문제는 다음과 같은 두 가지 이유로 어려운 문제이다. 첫번째 이유는 제안된 알고리즘의 검색 비용이 질의 시퀀스의 길이와 질의 윈도우의 길이 간의 차이에 따라 대체로 비례하여 증가하지만, 비례 기울기(slope)와 y-절편(y-intersect)은 질의 윈도우의 길이에 따라 가변적이다. 두번째 이유는 본 논문에서 질의 시퀀스 길이가 특정 분포를 따르지 않으며, 동일한 길이를 가지는 질의 시퀀스가 임의의 개수 존재할 수 있음을 가정하고 있다는 점이다. 본 논문에서는 최적의 (k - 1) 개의 질의 시퀀스 길이를 선택하기 위하여 근사 방법(approximation)에 기반한 알고리즘을 제안한다. 먼저, 임의로 선택된 몇 개의 윈도우 길이에 따라 인덱스를 생성하여 실험을 통하여 검색 비용을 측정한다. 다음에, 실험으로 얻어진 검색 비용을 이용하여 근사 방법을 통하여 임의의 윈도우 길이에 따라 생성된 인덱스를 이용할 때의 검색 비용을 추정(estimate)하는 공식을 생성한다. 근사 방법으로 보간법(interpolation)[16]을 이용한다. 마지막으로, 검색 비용 추정 공식(search cost estimation formula)에 기반하여 검색 성능을 최대화하도록 인덱스를 생성할 수 있는 (k - 1) 개의 질의 시퀀스 길이를 선택한다.

그림 6(a)는 길이 w_0와 w_1의 윈도우로 생성된 인덱스 I_0와 I_1을 이용할 때, 질의 시퀀스의 길이에 따른 제안된 검색 알고리즘의 검색 비용 그래프 C_0와 C_1을 보이고 있다. 인덱스 I_1은 w_1보다 짧은 길이의 질의 시퀀스를 찾기 위하여 사용할 수 없으므로 그래프 C_1은 w_1 이상 길이의 질의 시퀀스에 대해서만 나타나 있다. 그림 6(a)에서 인덱스 I_0만을 이용하여 제안된 검색 알고리즘이 모든 질의 시퀀스를 찾기 위한 비용은 그래프 C_0 아래의 영역(영역 A + B + C)의 면적과 같다. 이때, 제안된 알고리즘에 의하여 각 질의 시퀀스가 반환될 확률은 균일 분포를 따른다고 가정한다. 길이가 w_1 이상의 질의 시퀀스를 찾을 때에 인덱스 I_0를 이용할 수도 있지만, 인덱스 I_1을 이용할 때의 비용이 더 적다. 이는 인덱스 I_1을 이용하면 그룹 G_1 내의 질의 시퀀스의 길이와 윈도우의 길이 w_1과의 차이가 더 작아서 그만큼 착오 채택이 줄어들기 때문이다. 따라서, 그림 6(a)에서 인덱스 I_0와 I_1을 이용하여 제안된 알고리즘이 모든 질의 시퀀스를 찾기 위한 비용은 그래프 C_0와 C_1 아래의 영역(영역 A + C)의 면적과 같다. 길이 w_1 이상인 질의 시퀀스에 대하여 인덱스 I_1을 이용하는 경우와 인덱스 I_0를 이용하는 경우의 검색 비용 비율은 (영역 C의 면적) / (영역 B + C의 면적)과 같다.

특정 길이 w_j ($w_0 \le w_j \le L$)의 윈도우에 대해 생성된 인덱스 I_j를 이용하여 검색을 수행할 때의 검색 비용을 다음과 같이 근사적으로 구한다. 길이 w ($w_j \le w \le L$)의 질의 시퀀스를 찾기 위하여 제안된 검색 알고리즘이 인덱스 I_0를 이용한 비용을 $C_0(w)$라 하고, 인덱

스 I_j를 이용한 비용을 $C_j(w)$라 하자. 이때, 비용 $C_0(w)$ 및 $C_j(w)$는 실험을 통하여 얻어진다. 길이가 w_j 이상인 각각의 질의 시퀀스에 대하여 그 질의 시퀀스를 찾기 위한 검색 비용의 비율 $C_j(w)/C_0(w)$를 구하여 평균을 구한다. 이 평균 값을 인덱스 I_j를 이용할 때의 **검색 비용 비율**(*search cost ratio*)이라 부른다. 즉, 인덱스 I_j를 이용할 때의 검색 비용 비율 R_j는 아래의 공식 (7)과 같이 구해진다:

$$R_j = \frac{\sum_{w=w_j..L} C_j(w)/C_0(w)}{L-w_j+1} \quad (7)$$

여기에서, L은 최대 질의 시퀀스 길이이다. 그림 6(a)에서 R_1은 w_1 이상 길이 w의 모든 질의 시퀀스에 대하여 $C_1(w)/C_0(w)$ 비율 값들을 구하여 평균을 구함으로써 얻을 수 있다.

공식 (7)을 통하여 하나의 특정 w_j에 대하여 실험적으로 R_j를 구할 수 있다. 동일한 방법으로 $(d + 1)$ (d $\geq$ 0) 개의 w_j $(0 \leq j \leq d)$에 대하여 이러한 R_j를 구하면, 다음과 같이 근사적으로 임의의 w $(w_0 \leq w \leq L)$에 대하여 검색 비용 비율을 추정한다[1]. 검색 비용 비율은 그림 6(b)에서 보는 바와 같이 윈도우의 길이 w에 대해 유연하게 변화하므로, 아래의 공식 (8)과 같은 다항식(polynomial)으로 추정할 수 있다:

$$\hat{R}(w) = c_0 + c_1 w + c_2 w^2 + \cdots + c_d w^d. \quad (8)$$

여기에서, $\hat{R}(w)$는 **검색 비용 추정 비율**(*estimated search cost ratio*)이라 부르고 $(0.0 \leq \hat{R}(w) \leq 1.0)$, c_0, ..., c_d는 상수이다. 공식 (8)은 $w = w_j$ 이상 길이의 질의 시퀀스를 찾을 때에 제안된 알고리즘에서 인덱스 I_j를 이용할 때의 비용을 인덱스 I_0를 이용할 때의 비용으로 나눈 비율을 추정하기 위한 공식이다. 상수 c_0, ..., c_d의 값은 응용에 따라 달라질 것이며, 보간법[16]을 통하여 구할 수 있다.[2]

위의 공식 (8)은 질의 시퀀스의 길이가 균일 분포를 따르며 제안된 알고리즘이 각 질의 시퀀스를 반환하는 확률이 동일하다는 가정 하에서 구한 것이다. 하지만, 공식 (8)은 동일한 길이의 질의 시퀀스가 임의의 개수 존재하거나 제안된 알고리즘이 각 질의 시퀀스를 반환하는 확률이 동일하지 않더라도 유용하다. 아래의 보조 정리 1과 2는 그러한 사실을 증명한다.

보조 정리 1. 공식 (8)은 동일한 길이의 질의 시퀀스가 임의의 개수 존재하여도 변경 없이 사용 가능하다.

증명. 공식 (7)에서 w $(w_j \leq w \leq L)$ 길이를 갖는 질의 시퀀스의 빈도(frequency)를 f_w (≥ 0)라고 하자. 제안된 알고리즘이 인덱스 I_j와 인덱스 I_0를 이용하여 길이 w의 질의 시퀀스를 모두 찾기 위한 비용은 각각 $C_j(w)\cdot f_w$ 와 $C_0(w)\cdot f_w$ 이다. 따라서, 검색 비용 비율의 평균을 구하면 다음과 같다:

$$R_j^f = \frac{\sum_{w=w_j..L}(C_j(w)\cdot f_w)/(C_0(w)\cdot f_w)}{L-w_j+1} = \frac{\sum_{w=w_j..L} C_j(w)/C_0(w)}{L-w_j+1} = R_j.$$

공식 (8)의 $\hat{R}(w)$는 R_j를 이용하여 구해지며 R_j^f가 R_j와 같으므로, R_j^f를 이용하더라도 공식 (8)을 그대로 변경 없이 사용할 수 있다. □

보조 정리 2. 공식 (8)은 제안된 알고리즘이 각 질의 시퀀스를 반환하는 확률이 동일하지 않더라도 변경 없이 사용 가능하다.

증명. 보조 정리 1과 같은 방법으로 증명한다. 공식 (7)에서 w $(w_j \leq w \leq L)$ 길이를 갖는 질의 시퀀스가 제안된 알고리즘에 의해 반환되는 확률(probability)을 p_w $(0.0 \leq p_w \leq 1.0)$라고 하자. 제안된 알고리즘이 인덱스 I_j와 인덱스 I_0를 이용하여 길이 w의 질의 시퀀스를 반환하기 위한 비용은 각각 $C_j(w)\cdot p_w$와 $C_0(w)\cdot p_w$이다. 따라서, 검색 비용 비율의 평균을 구하면 다음과 같다:

$$R_j^p = \frac{\sum_{w=w_j..L}(C_j(w)\cdot p_w)/(C_0(w)\cdot p_w)}{L-w_j+1} = \frac{\sum_{w=w_j..L} C_j(w)/C_0(w)}{L-w_j+1} = R_j.$$

공식 (8)의 $\hat{R}(w)$는 R_j를 이용하여 구해지며 R_j^p가 R_j와 같으므로, R_j^p를 이용하더라도 공식 (8)을 그대로 변경 없이 사용할 수 있다. □

공식 (8)은 인덱스 I_0를 이용한 검색 비용에 대비한 비율 공식이므로, 인덱스 I_0를 이용한 검색 비용은 실험적으로 구해야 한다. 검색 비용 추정 비율 공식 $\hat{R}(w)$와 인덱스 I_0를 이용한 검색 비용이 구해지면, 전체 검색 비용을 최소화하고 검색 성능을 최대화하기 위한 $(k - 1)$ 개의 윈도우의 길이 w_1, ..., w_{k-1}를 구할 수 있다. 그림 9는 최적의 $(k - 1)$ 개의 윈도우 길이를 구하기 위한 휴리스틱(heuristic) 알고리즘을 보인 것이다. 여기에서, $w_k = L + 1$이라고 가정한다.

그림 9의 라인 (2)에서 전체 검색 비용을 구하기 위한 데이타 구조로 인덱스 I_0를 이용한 검색 비용을 누적하여(cumulative) 저장하기 위한 하나의 배열(array) $CC_0[w_0..L]$이 필요하다. 이 배열의 하나의 요소(element) $CC_0[w]$ $(w_0 \leq w \leq L)$에는 인덱스 I_0를 이용

---

1) 여기에서, d 값은 공식 (8)의 다항식의 차수(degree)라고 하며, 인덱스의 개수 k보다 클 수도 있고 작을 수도 있다.

2) 실제적으로, 공식 (8)의 다항식의 차수는 d보다 작은 d' ($< d$)이 될 수 있다. 또한, 특정 w ($w_0 \leq w \leq L$)에 대한 검색 비용 추정 비율 $\hat{R}(w)$를 구할 때에 공식 (8)의 다항식을 사용하지 않고 w 근처의 d'' ($< d$) 개의 w_j $(0 \leq j \leq d)$에 대한 R_j 값만을 이용하여 직접 구할 수도 있다 [16]. 대개 d' 및 d'' 값은 3~6 범위의 고정된 작은 상수 값이다.

(1) Evenly distribute w_j $(1 \le j < k)$.
    That is, for each j, set $w_j = w_0 + (L - w_0) \cdot (j / k)$;
(2) Compute the overall search cost;
(3) **repeat**
(4) **for** j = 1 to k–1 **do**
(5) Set w_j as the value in the range (w_{j-1}, w_{j+1})
        such that the overall search cost becomes minimum;
(6) **end for**
(7) **until** no change.

그림 9 최적의 윈도우 길이를 찾기 위한 알고리즘

하여 제안된 알고리즘이 길이 w_0에서 w까지의 질의 시퀀스를 검색하기 위한 비용이 저장된다. 즉, 아래의 공식 (9)와 같은 값이 저장된다:

$$CC_0[w] = \sum_{i=w_0}^{w} C_0(i) \quad (9)$$

그림 6(a)에서 $CC_0[w_1]$의 값은 영역 A의 면적과 같다. 배열 CC_0의 w-번째 요소 값을 구하기 위하여 $w = w_0$로부터 순차적으로 $(w - 1)$-번째 요소 값을 이용하여 구하면, 배열 CC_0를 구성하기 위한 시간 복잡도는 $O(n)$이다$(n = L - w_0 + 1)$.

전체 검색 비용을 구하는 방법은 다음과 같다. 먼저, 각 j에 대하여 제안된 알고리즘이 인덱스 I_0를 이용하여 검색을 수행할 때의 비용을 구한다. 이 비용은 위에서 구성한 배열 CC_0을 이용하여 쉽게 구할 수 있다. 이때의 비용을 인덱스 I_j를 이용하여 검색을 수행할 때의 비용으로 환산한다. 이 비용은 공식 (8)의 검색 비용 추정 비율 공식 $\hat{R}(w)$를 이용하여 구한다. 전체 검색 비용은 각 j에 대하여 구해진 비용을 모두 더한 값이 되며, 공식으로 정리하면 다음과 같다:

$$SC = \sum_{j=0}^{k-1} \hat{R}(w_j) \cdot (CC_0[w_{j+1} - 1] - CC_0[w_j]) \quad (10)$$

여기에서, $w_k = L + 1$이며 $CC_0[w_0] = 0$, $\hat{R}(w_0) = 1.0$이라 가정한다. 공식 (10)을 이용하여 전체 검색 비용을 구하기 위한 시간 복잡도는 $O(k)$이다.[3)]

그림 9의 알고리즘의 시간 복잡도는 다음과 같다. 라인 (5)에서 w_j에 대하여 w_{j-1}과 w_{j+1} 사이의 모든 윈도우 길이를 대입하여 전체 검색 비용을 구한다. 이때, 공식 (10)을 이용하여 전체 검색 비용을 새로 구하지 않고 w_j 값의 변동에 의한 검색 비용의 변경만을 반영해 주므로, 라인 (5)의 시간 복잡도는 $O(n)$이다. 라인 (4)~(6)의 for 루프는 k 번 실행되므로 시간 복잡도는 $O(kn)$이다. 라인 (3)~(7)의 repeat 루프가 r 번 수행된다고 하면, 알고리즘 전체의 시간 복잡도는 $O(rkn)$이다. 최적의 윈도우 길이 w_0, ..., w_{k-1}을 찾기 위하여 동적 프로그래밍 (dynamic programming) 알고리즘을 사용할 수도 있으나, 이때의 시간 복잡도는 $O(kn^2)$이다[17]. 본 논문에서의 실험 결과 repeat 루프를 실행하는 횟수 r은 작은 수로 한정되어 있었다($r << n$). 따라서, 본 논문에서 제안한 휴리스틱 알고리즘은 동적 프로그래밍 알고리즘에 비하여 효율적이라 판단할 수 있다.

3) 만약 $\hat{R}(w)$를 구하기 위한 w 값의 개수가 많지 않다면 모든 w에 대하여 미리 $\hat{R}(w)$ 값을 구하여 하나의 배열에 저장해 놓을 수도 있다.

6. 알고리즘의 재차 확장

참고문헌 [8]에서는 연속성 개념을 제시하였다. 연속성이란 현재 시점 t_0의 스트리밍 서브시퀀스 $X^{(t_0)}$는 바로 다음 시점 $(t_0 + 1)$의 스트리밍 서브시퀀스 $X^{(t_0+1)}$과 거의 같으며, 이들을 이용한 검색 결과도 유사하다는 점이다. 즉, 스트리밍 서브시퀀스 $X^{(t_0)}$와 $X^{(t_0+1)}$을 이용한 검색 결과의 공통되는 비율(%)이 매우 높다는 것이다. 참고문헌 [8]에서는 이러한 연속성을 이용하여 가까운 미래의 스트리밍 시계열에 대한 질의 결과를 미리 예측하여 검색하는 알고리즘을 제시하였고, 연속성을 이용하지 않을 때에 비하여 검색 성능을 향상시킬 수 있음을 보였다. 본 절에서는 이러한 연속성의 개념에 기반하여 검색 성능을 보다 향상시킬 수 있도록 제4절에서 제시한 알고리즘을 재차 확장한다. 정규화 변환을 지원하는 스트리밍 서브시퀀스 매칭에서는 스트리밍 서브시퀀스와 질의 시퀀스를 비교하기 전에 정규화 변환을 먼저 실행하므로, 이러한 연속성이 기존에 비하여 덜 성립한다. 따라서, 정규화 변환을 지원하는 스트리밍 서브시퀀스 매칭에 있어서 연속성을 활용하기 위한 새로운 방안이 필요하다.

재차 확장된 알고리즘의 기본 아이디어는 현재 시점에서 연속성 개념을 이용하여 앞으로 일정 시간 동안의 후보 질의 시퀀스를 한번의 인덱스 검색만으로 찾아낸다는 것이다. 그렇게 함으로써 일정 시간 동안의 인덱스

검색을 위한 비용을 줄이게 되고 궁극적으로 검색 성능을 향상시킬 수 있다. 그림 10은 제4절에서 제시한 알고리즘의 재차 확장에 대한 기본적인 아이디어를 보인 것이다. 앞에서 제시한 알고리즘에서는 스트리밍 윈도우 X_w에 대하여 모든 질의 시퀀스 내의 최우측 윈도우와만 유사성을 비교하였다. 즉, 그림 10에서 질의 시퀀스 Q_0 내의 윈도우 $Q_{0\text{-}w}$와만 비교하였다. 재차 확장된 알고리즘에서는 스트리밍 윈도우 X_w에 대하여 질의 시퀀스 내의 최우측 윈도우뿐만 아니라 최우측으로부터 오프셋이 1, ..., m - 1인 윈도우와도 유사성을 비교한다. 만약 윈도우 X_w가 질의 시퀀스 Q 내의 우측 오프셋이 f ($0 \le f < m$)인 윈도우와 유사하다면, 질의 시퀀스 Q는 시점 ($t_0 + f$)의 스트리밍 서브시퀀스 $X^{(t_0+f)}$와 유사한 질의 시퀀스가 될 수 있다. 예를 들어, 그림 10에서 윈도우 X_w와 질의 시퀀스 내의 우측 오프셋이 1인 윈도우 $Q_{1\text{-}w}$와 유사하다면, 질의 시퀀스 Q_1은 스트리밍 서브시퀀스 $X^{(t_0+1)}$과 유사한 질의 시퀀스가 될 수 있다. 시점 (t_0 + 1)에 질의 시퀀스 Q_1과 스트리밍 서브시퀀스 $X^{(t_0+1)}$을 정규화 변환한 후의 실제 거리가 ε보다 작다면 두 시퀀스는 유사하다고 판단하며 질의 시퀀스 Q_1을 시점 (t_0 + 1)에 반환한다.

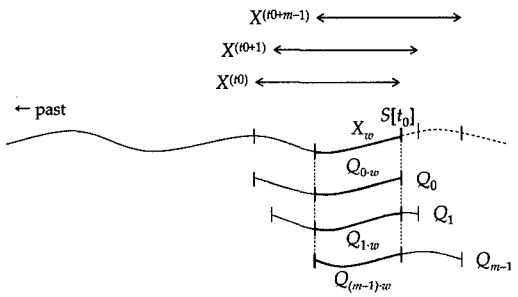


그림 10 재차 확장된 알고리즘

그림 11은 재차 확장된 알고리즘의 인덱싱 단계와 검색 단계를 정리한 것이다. 인덱싱 단계의 라인 (1)에서 인덱스를 생성할 최소 윈도우 길이 w_0는 앞 절에서 설명한 것과 달리 (최소 질의 시퀀스 길이 - m)이 되어야 한다. 라인 (4)에서 각 질의 시퀀스에 대하여 최우측 오프셋이 f ($0 \le f < m$)인 모든 윈도우를 추출하여 인덱스를 생성한다. 이때, 각 윈도우에 대하여 오프셋 f를 함께 인덱스에 저장한다. 검색 단계 알고리즘은 제4절에서 제시한 알고리즘과 달리, 한번의 인덱스 검색만으로 현재 시점 t_0부터 미래 시점 (t_0 + m - 1)까지의 스트리밍 서브시퀀스에 대한 검색을 수행한다. 라인 (3)에서 하나의 스트리밍 윈도우 X_w에 대하여 인덱스를 검색하면 질의 시퀀스 내의 최우측 오프셋에 관계 없이 윈도우 X_w와 유사한 모든 질의 윈도우를 찾게 된다. 인덱스로부터 찾은 모든 질의 윈도우는 같은 오프셋 f에 따라 그룹으로 나눈다. 이러한 작업을 효율적으로 하는 방법

(1) Choose k window sizes w_0, ..., w_{k-1};
(2) Divide query sequences into k groups G_j ($0 \le j < k$) using Eq. (6);
(3) **for** each group G_j **do**
(4) Extract windows Q_w 's of length w_j whose right offsets are 0, ..., m–1 from Q;
(5) Do the same as steps (3) ~ (4) in the indexing phase
in Simple Algorithm (Fig. 5(a));
(6) **end for.**

(a) 인덱싱 단계

(1) **for** each group G_j ($0 \le j < k$) **do**
(2) Extract rightmost window X_w of length w_j from streaming time-series S;
(3) Do the same as steps (2) ~ (3) in the searching phase
in Simple Algorithm (Fig. 5(b)) using the index I_j;
(4) **for** f = 0 to m – 1 **do**
(5) For each window Q_w whose right offset is f, do the same as steps (5) ~ (7)
in the searching phase in Simple Algorithm (Fig. 5(b));
(6) **end for**
(7) **end for.**

(b) 검색 단계

그림 11 재차 확장된 알고리즘의 인덱싱 및 검색 단계

은 미리 f 개의 연결 리스트(linked list)로 구성된 하나의 배열을 준비하고 질의 윈도우를 인덱스에서 검색해 가면서 오프셋 f에 따라 해당 연결 리스트의 마지막에 연결하는 것이다. 인덱스를 검색해 얻어진 모든 질의 윈도우 Q_w와 함께 오프셋 f와 스트리밍 윈도우 X_w로부터의 거리 e (≥ 0)가 관리된다. 라인 (4)~(6)의 for 루프에서는 질의 윈도우의 오프셋에 따라 순차적으로 처리한다. 즉, 시점 $(t_0 + f)$에서는 오프셋이 f인 윈도우를 포함하는 질의 시퀀스만을 고려하며 스트리밍 서브시퀀스 $X^{(t_0+f)}$와 유사한 질의 시퀀스를 찾는다.

재차 확장된 알고리즘은 앞으로 입력될 스트리밍 시계열에 대한 유사성 검색을 미리 수행하는 것이므로, 공식 (4)를 이용하여 ε'을 계산할 때에 현재 입력되지 않은 스트리밍 서브시퀀스 X에 대한 $\sigma(X)$를 미리 알 수 없다는 문제가 있다. 따라서, $\sigma(X)$에 대한 추정값 $\hat{\sigma}(X)$를 구하여 ε'의 추정값 $\hat{\varepsilon}'$을 계산한다. 추정 표준편차 $\hat{\sigma}(X)$에 대한 하나의 후보로 과거의 일정 기간 동안의 X와 같은 길이의 모든 스트리밍 서브시퀀스의 표준편차 중의 최대값을 사용할 수 있다. 이렇게 하여 구해진 추정 한계값 $\hat{\varepsilon}'$을 이용하여 검색을 수행한 다음에, 각 시점 t $(t_0 \leq t < t_0 + m - 1)$에 입력된 스트리밍 서브시퀀스 X에 대한 실제의 ε'을 구하여 $\hat{\varepsilon}'$과 비교한다. 만약 $\hat{\varepsilon}' \geq \varepsilon'$이라면, 스트리밍 윈도우 X_w로부터의 거리 e가 ε'보다 큰 질의 윈도우 Q_w가 반드시 존재하며, 그러한 질의 윈도우에 대해서는 그림 11(b)의 검색 단계의 라인 (5)를 수행하지 않고 그냥 버리면 된다. 만약 시점 t에 $\hat{\varepsilon}' < \varepsilon'$이라면, 인덱스를 이용하여 스트리밍 윈도우 X_w로부터의 거리 e가 $\hat{\varepsilon}' \leq e < \varepsilon'$인 질의 윈도우 Q_w만을 검색한다. 재차 확장된 알고리즘은 제4절에서 설명한 두 알고리즘과 같이 착오 기각이 발생하지 않음을 쉽게 보일 수 있다.

본 절에서 설명한 재차 확장된 알고리즘은 제4.2절에서 설명한 알고리즘과 마찬가지로 기본적으로 정규화 변환을 수행함을 가정하고 있다. 즉, 미래 시점의 후보 질의 시퀀스를 미리 검색 가능한 근거도 공식 (4)에서 스트리밍 시퀀스 X 내의 스트리밍 윈도우 X_w의 오프셋 f와 무관하게 새로운 검색 범위 ε'을 구할 수 있기 때문이다. 이러한 가정은 정규화 변환을 수행하지 않는 다른 시계열 매칭 알고리즘에는 일반적으로 적용 가능하지 않다.

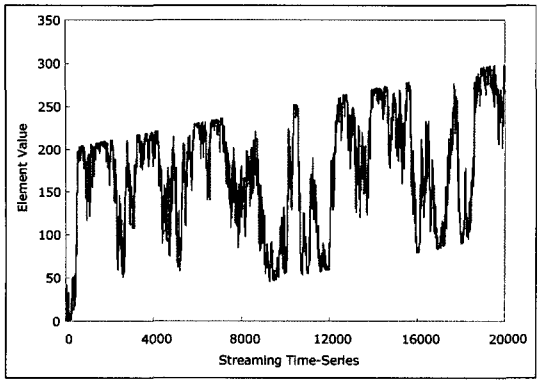
7. 성능 평가

본 절에서는 본 논문에서 제안된 알고리즘들의 성능을 평가한다. 제7.1절에서는 실험 설정 및 데이타에 대해 설명한다. 제7.2절에서는 그림 6에서 보인 검색 비용 경향을 실험적으로 확인하고, 제7.3절에서는 제안된 알고리즘들의 성능 평가 결과를 설명한다.

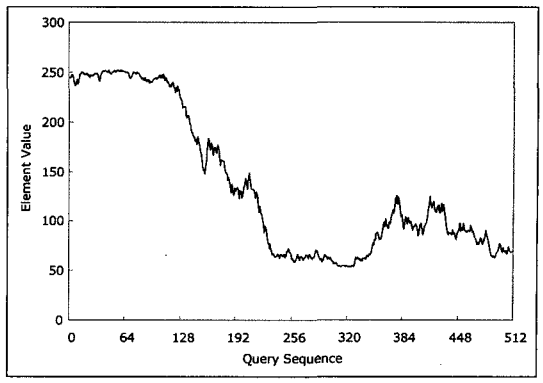
7.1 실험 환경

본 실험에서는 Linux 운영체제 환경에서 스트리밍 환경을 모의 실험하기 위하여 네임드 파이프(named pipe)[18]로 연결된 두 가지의 프로세스를 구현하였다. 한 가지 프로세스는 스트리밍 시계열을 생성하는 데이타 프로세스이고, 다른 한 가지 프로세스는 스트리밍 서브시퀀스 매칭을 수행하는 검색 프로세스이다. 데이타 프로세스에서는 연속적으로 스트리밍 시계열 요소 값을 생성하거나 파일로부터 읽어 들여 네임드 파이프를 통해 검색 프로세스로 전송하고, 검색 프로세스에서는 입력된 스트리밍 시계열 요소 값에 따라 실제 검색을 수행한다. 검색 프로세스는 수행하고자 하는 검색 알고리즘에 따라 별도로 구현된다. 본 절에서는 순차검색 알고리즘을 *SeqScan*으로, 제4절에서 제시한 간단한 알고리즘과 확장된 알고리즘을 각각 *SingleIdx*, *MultiIdx*로 표기한다. 확장된 알고리즘은 인덱스를 생성하기 위한 윈도우 길이를 구하는 방법에 따라 다시 구분되며, 균등 분포에 의한 방법, 제5절에서 제시한 휴리스틱에 의한 방법, 동적 프로그래밍에 의한 방법에 따른 알고리즘을 각각 *Multi_Even*, *Multi_Heur*, *Multi_Dyn*으로 표기한다.

실험을 수행하기 위한 하드웨어 플랫폼은 Intel Pentium 4 2.8GHz CPU, 512MB RAM, 80GB 하드 디스크를 장착한 PC이며, 소프트웨어 플랫폼은 GNU/Linux Version 2.6.6 운영체제이다. *SeqScan* 이외의 다른 검색 프로세스는 다차원 인덱스로 R*-트리[14]를 사용하였다. DFT를 통한 저차원 변환을 위하여 사용한 특성값의 개수는 $f = 6$으로 하였다. 본 실험에서 스트리밍 시계열은 합성 데이타(synthetic data)를 사용하였다. 스트리밍 시계열은 참고문헌 [7]에서와 같이 $S[i] = 100 \cdot [\sin(0.1 \cdot RandomWalk[i]) + 1.0 + i/20000]$ $(i = 0 .. 19999)$를 사용하여 생성하였다. 여기에서, 랜덤 워크(random walk) 시리즈 $RandomWalk[i]$는 최초 값을 1500.0으로 하고, 다음 값 $RandomWalk[i+1]$은 바로 이전 값 $RandomWalk[i]$에 (-1.0, 1.0) 범위의 임의의 값을 더하여 생성하였다. 질의 시퀀스는 참고문헌 [7]에서와 같이 스트리밍 시계열의 임의 위치로부터 길이 w인 서브시퀀스를 추출하여 사용하였으며, 질의 시퀀스 길이 w는 128부터 32씩 증가시키면서 1024까지, 즉 $w = 128, 160, ..., 1024$를 사용하였다. 그림 12(a)는 스트리밍 시계열을, 그림 12(b)는 길이 512인 질의 시퀀스의 예를 보인 것이다.



(a) 스트리밍 시계열



(b) 하나의 질의 시퀀스

그림 12 실험 데이타

본 실험에서는 질의 시퀀스 길이의 분포에 따라 네 개의 질의 집합을 생성하였다. 첫번째 질의 집합은 $\boldsymbol{Q}_{even}$으로 표기하고, 길이에 관계없이 같은 개수(60개씩)의 질의 시퀀스가 포함된 집합이다. 두번째부터 네번째 질의 집합은 $\boldsymbol{Q}_1$, $\boldsymbol{Q}_2$, $\boldsymbol{Q}_4$로 표기하고, 각 길이 w에 따른 질의 시퀀스의 개수는 아래의 공식에 의하여 정해진다:

$$QC(w) = Amp \cdot \left[\cos\left(Freq \cdot \pi \cdot \frac{w - w_0}{L - w_0} \right) + 1.0 \right] + Min \quad (11)$$

여기에서, $QC(w)$는 길이 w인 질의 시퀀스의 개수, Amp, $Freq$, Min은 각각 진폭(amplitude), 주기(frequency), 최소값(minimum)을 의미하고, w_0 = 128, L = 1024이다. 본 실험에서는 Amp = 50, Min = 10으로 설정하였고, 질의 집합 $\boldsymbol{Q}_1$, $\boldsymbol{Q}_2$, $\boldsymbol{Q}_4$는 각각 Freq = 1, 2, 4로 설정하여 질의 시퀀스 개수를 구하였다. 질의 집합 $\boldsymbol{Q}_1$, $\boldsymbol{Q}_2$, $\boldsymbol{Q}_4$는 본 논문에서 제안된 알고리즘들이 임의 분포의 질의 시퀀스 길이에 대해서도 효율적인 검색을 수행함을 보이기 위한 것이다. 이와 같이 구성된 질의 시퀀스들은 스트리밍 시계열의 특징을 그대로 따른다. 이는 유사성 검색에서의 질의 시퀀스는 일반적으로 검색 대상이 되는 데이타 시퀀스[2,5] 또는 스트리밍 시계열[7]과 유사한 특징을 가지기 때문이다. 본 실험에서는 참고문헌 [7]에서와 같이 하나의 질의 집합 내의 모든 질의 시퀀스가 메인 메모리 상에 적재되어 있음을 가정한다.

7.2 검색 비용 실험 결과

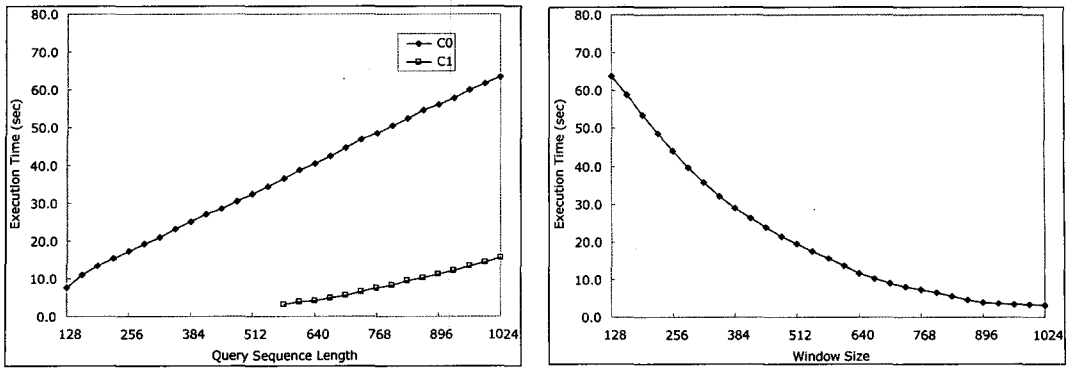
본 절에서는 그림 6에서 보인 검색 비용 경향을 실험적으로 확인한다. 본 절에서의 실험을 위하여 윈도우 길이 w_0 = 128, w_1 = 576, 검색 한계 ε = 7.5로 설정하였고 질의 집합 $\boldsymbol{Q}_{even}$을 사용하였다. 검색 한계 ε이 일정하여도 질의 시퀀스의 길이에 따라 선택률(selectivity)은 변화한다. 이때, 선택률은 질의 시퀀스의 길이가 길어짐에 따라 급격히 감소하며, 본 실험에서는 질의 시퀀스의 길이가 384 이상인 경우에 선택률이 대체로 10^{-3} 정도를 유지하였다.

그림 13은 질의 시퀀스 및 윈도우 길이에 따른 검색 비용에 대한 실험 결과를 보인 것이다. 그림 13(a)는 질의 시퀀스에 따른 검색 비용 그래프이고, w_1 = 576이므로 비용 그래프 C_1은 길이 576 이상의 질의 시퀀스에 대해서만 검색 비용을 나타낼 수 있다. 비용 그래프 C_0와 C_1 모두 질의 시퀀스의 길이가 증가함에 따라 검색 비용도 거의 선형으로 증가하였다. 그림 13(a)에서 보는 바와 같이, 길이 576 이상의 질의 시퀀스에 대해서는 인덱스 I_0보다 인덱스 I_1을 이용하는 것이 검색 비용을 줄이기 위하여 현명한 선택이다. 그림 13(b)는 윈도우 길이에 따른 검색 비용 그래프이다. 그림에서 보는 바와 같이, 윈도우 길이가 증가함에 따라 검색 비용이 유연하게 변화하므로 보간법을 이용하여 임의의 윈도우 길이에 따른 검색 비용을 추정하기가 용이하다. 특히, 공식 (8)에서 보인 다항식보다 훨씬 낮은 차수 d' ($< d$)의 다항식으로도 정확하게 (오차가 매우 작게) 검색 비용을 추정할 수 있다.

7.3 성능 비교 실험 결과

본 절에서는 먼저 제5절에서 제시한 휴리스틱 알고리즘과 동적 프로그래밍 알고리즘에 의하여 인덱스를 생성할 윈도우 길이가 어떻게 선택되는지 비교한다. 여기에서, 질의 집합은 앞에서 제시한 $\boldsymbol{Q}_{even}$ 및 $\boldsymbol{Q}_1$, $\boldsymbol{Q}_2$, $\boldsymbol{Q}_4$를 사용하고 인덱스의 개수는 k = 2, 3, 4, 5로 설정하였다. 윈도우 길이 선택 결과, k = 2, 3인 경우에는 두 알고리즘이 선택한 윈도우 길이가 동일하였고, k = 4인 경우에도 거의 유사하였다. 표 2는 k = 5인 경우 각 질의 집합에 대하여 두 알고리즘이 선택한 윈도우 길이를 보인 것이다. 표에서 보는 바와 같이, k = 5인 경우에도 두 알고리즘의 결과가 매우 유사함을 알 수 있다.

본 논문에서는 다섯 가지 실험을 수행한다. 첫번째 실



(a) 질의 시퀀스 길이에 따른 검색 비용 (b) 윈도우 길이에 따른 검색 비용

그림 13 질의 시퀀스 및 윈도우 길이에 따른 검색 비용 경향

표 2 윈도우 길이 선택 결과(k = 5)

Query Set	Algorithm	Window Sizes
Qeven	Heuristic	128, 256, 416, 576, 800
	Dynamic	128, 288, 448, 608, 800
Q1	Heuristic	128, 224, 320, 448, 640
	Dynamic	128, 256, 384, 512, 704
Q2	Heuristic	128, 224, 352, 640, 864
	Dynamic	128, 224, 352, 640, 864
Q4	Heuristic	128, 224, 416, 544, 864
	Dynamic	128, 256, 448, 576, 864

험에서는 인덱스의 개수 k에 따라서, 두번째 실험에서는 검색 한계 ε에 따라서 알고리즘들 간의 성능을 비교한다. 첫번째와 두번째 실험에서 성능 비교의 대상이 되는 알고리즘들은 *SeqScan, SingleIdx, Multi_Even, Multi_Heur, Multi_Dyn* 알고리즘이며, 네 개의 실험 집합에 대하여 개별적으로 실험하였다. 세번째 실험에서는 m 값에 따른 *SingleIdx* 알고리즘의 검색 성능의 변화를 관찰한다. 네번째 실험에서는 질의 시퀀스 개수를 달리하면서, 다섯번째 실험에서는 실제 주가 데이타를 사용하여 알고리즘들 간의 성능을 비교하였다.

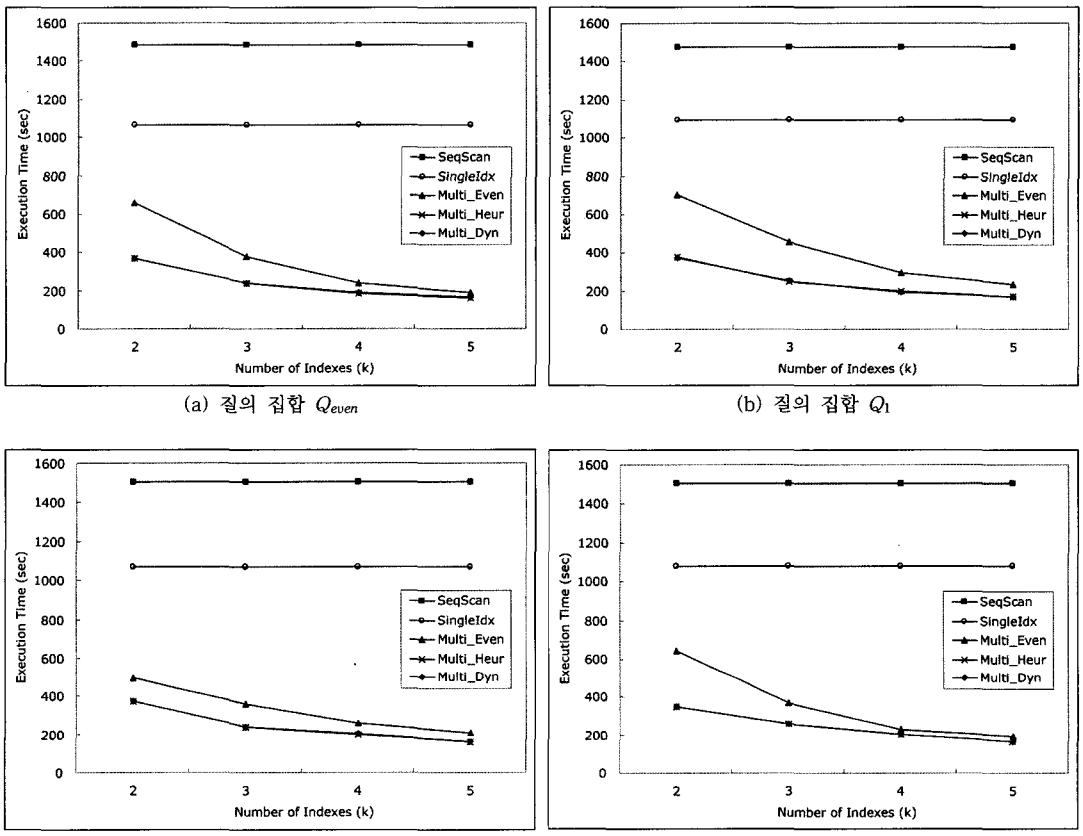
첫번째 실험에서는 ε = 3.5로 설정하고 인덱스의 개수 k = 2, 3, 4, 5에 대하여 성능을 비교하였다. 그림 14는 첫번째 실험의 결과를 보인 것이다. 가로 축은 인덱스의 개수 k, 세로 축은 알고리즘의 실행시간(execution time)을 나타낸다. 그림에서 보는 바와 같이, 모든 질의 집합에 있어서 제안된 알고리즘들은 순차 검색 *SeqScan* 알고리즘에 비하여 월등한 성능을 가지며, 다중 인덱스를 이용하는 *MultiIdx* 알고리즘들이 하나의 인덱스를 사용하는 *SingleIdx* 알고리즘에 비하여 훨씬 나은 성능을 가진다. 전체적으로 k가 증가함에 따라 성능이 향상되었다. 다중 인덱스 알고리즘들 중에서 *Multi_Heur*와 *Multi_Dyn* 알고리즘 간에는 성능 차이가 거의 없는 데에 반하여, *Multi_Even* 알고리즘에 비하여 향상된 검색 성능을 보이고 있다. 본 논문에서 제안된 *Multi_Heur* 알고리즘은 *SeqScan, SingleIdx, Multi_Even* 알고리즘들에 비하여 각각 최대 9.3 배, 6.7 배, 1.9 배 성능이 향상되었다. 반면에 *Multi_Dyn* 알고리즘과의 차이는 최대 2.7%에 불과하였다.

본 논문에서는 *SeqScan* 알고리즘의 거리 계산에 있어서 조기 포기(early abandoning) 기법[19]이나 DFT 변환을 사용하지 않았다. 그러한 방법들을 사용하여도 성능에 큰 차이가 없기 때문이며, 그 세부적인 이유는 다음과 같이 두 가지이다. 첫째, 본 논문의 유사성 매칭에서는 거리 계산보다 정규화 변환에 더 많은 CPU 연산이 필요하여 *SeqScan* 알고리즘에서 거리 계산을 일부 줄이는 것으로 큰 효과를 보지 못하기 때문이다. 둘째, 조기 포기 기법을 통해 거리 계산 오버헤드를 일부 줄일 수 있는 반면, 각 요소 값에 대한 거리 계산 이후에 ε^2와의 비교 연산이 추가 오버헤드로 작용하여 그 효과를 상쇄시키기 때문이다.[4] 실제로, 정규화 변환은 시퀀스들 간의 거리를 크게 감소시키는 역할을 하며, 조기 포기 기법이 그러한 상황에서 큰 효과를 거두지 못함은 알려져 있다[19].

두번째 실험에서는 k = 3으로 설정하고 검색 한계 ε = 1.5, 3.5, 5.5, 7.5에 대하여 성능을 비교하였다.[5] 그림 15는 두번째 실험의 결과를 보인 것이다. 가로 축은 검색 한계 ε, 세로 축은 알고리즘의 실행시간을 나타낸다. 전체적으로 ε이 감소함에 따라 성능이 향상되었다. 검색 한계 ε = 7.5에 대하여 *SingleIdx* 알고리즘이 *SeqScan* 알고리즘보다 저하된 성능을 갖는 원인은 ε이 커짐에

4) 본 논문의 실험에서는 일반적인 상황에 대처하기 위하여 정수(int)가 아닌 실수(double) 변수로 ε 및 시퀀스간 거리를 표현한다.

5) 실험 결과, 각 검색 한계에 대한 평균 선택률(selectivity)[2,4,5]은 0.03%, 0.10%, 0.45%, 1.89%로 나타났다. 실제의 응용에서 매칭을 수행할 때에는 선택률이 아닌 검색 한계를 사용하므로, 본 논문에서는 선택률 대신에 검색 한계를 변수로 실험 결과를 제시한다.



(a) 질의 집합 Q_{even} (b) 질의 집합 Q_1

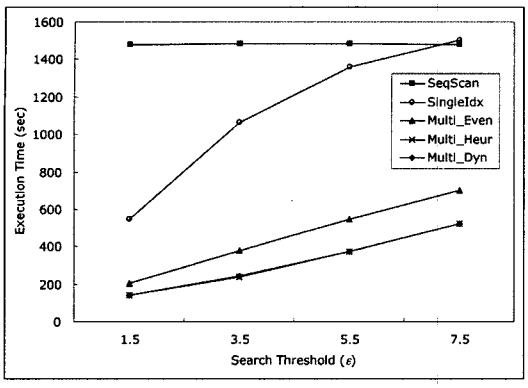
(c) 질의 집합 Q_2 (d) 질의 집합 Q_4

그림 14 첫번째 실험 결과

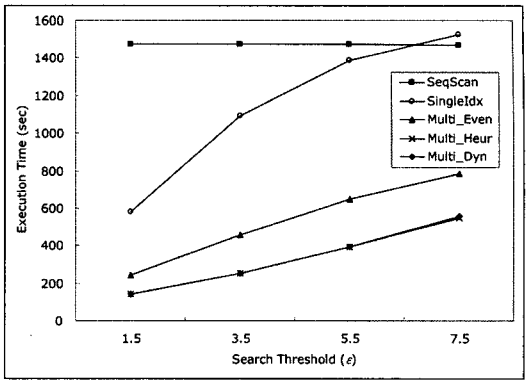
따라 착오 채택이 급격히 늘어남과 동시에 인덱스 자체를 검색하기 위한 비용도 크게 늘어나기 때문이다. 본 논문에서 제안된 *Multi_Heur* 알고리즘은 *SeqScan*, *SingleIdx*, *Multi_Even* 알고리즘들에 비하여 각각 최대 13.2 배, 4.5 배, 1.8 배 성능이 향상되었다. 반면에 *Multi_Dyn* 알고리즘과 비교하여 최대 30.7% 성능이 향상되었으며, *Multi_Dyn* 알고리즘보다 낮은 성능을 보이는 경우에는 최대 2.4% 차이에 불과하였다.

세번째 실험에서는 k = 1, ε = 3.5로 설정하고 m = 4, 8, ..., 64로 변경하면서 검색 성능의 변화를 관찰하였다. 이 실험에서 수행한 알고리즘은 *SingleIdx* 알고리즘이며, m 값에 따라 *SingleIdx*(m)으로 표기한다. 이 실험은 질의 집합 $\boldsymbol{Q}_{even}$과 $\boldsymbol{Q}_1$에 대하여 질의 시퀀스 길이의 범위를 변경하면서 수행하였다. 즉, 길이가 256~1024, 512~1024, 512~768인 질의 시퀀스만으로 구성된 질의 집합을 대상으로 실험하였다. 그림 16은 세번째 실험의 결과를 보인 것이다. 가로 축은 m 값, 세로 축은 *SingleIdx*(1) 알고리즘의 실행시간을 *SingleIdx*(m) 알고리즘에 의한 실행시간으로 나눈 비율을 나타낸다. 그림에서 보는 바와 같이, m 값이 증가하면서 검색 성능 비율이 급격히 증가하다가 서서히 감소하는 경향을 보이고 있다. 이러한 현상의 원인은 인덱스를 한번만 검색함으로써 검색 성능이 증가하나, m 값이 특정 한계를 넘어서면 예측에 의한 착오 채택이 점차 증가하기 때문이다. 그림에서, 질의 시퀀스의 길이 범위가 작을수록 성능이 더욱 향상하였다. 그 원인은 최대 질의 시퀀스 길이 L과 인덱스를 생성한 윈도우 길이 w_0 간의 크기가 점차 감소하기 때문이며, k 값을 증가함에 따라 검색 성능을 향상시킬 수 있다. 세번째 실험에서 *SingleIdx*(m) 알고리즘은 *SingleIdx*(1)에 비하여 최대 2.76 배까지 검색 성능이 향상되었다. 인덱스의 개수 k가 증가함에 따라 재차 확장된 알고리즘의 검색 성능은 더욱 향상될 것으로 판단된다.

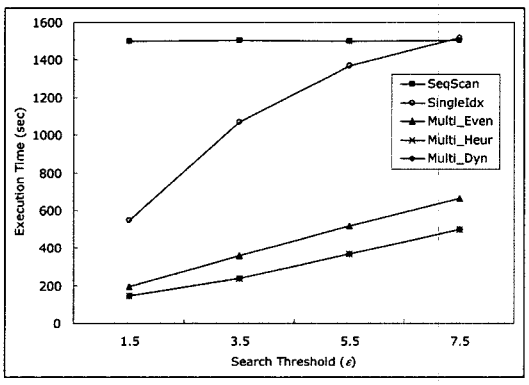
네번째 실험에서는 질의 시퀀스 개수를 달리하면서 제안된 알고리즘의 우수성을 확인하였다. 질의 시퀀스의 생성은 $\boldsymbol{Q}_{even}$과 동일한 방법을 사용하였으며, 각 질의 시퀀스 길이에 대한 개수를 6 개(전체 174 개), 60 개(전체 1,740 개), 600 개(전체 17,400 개)로 변경하면서



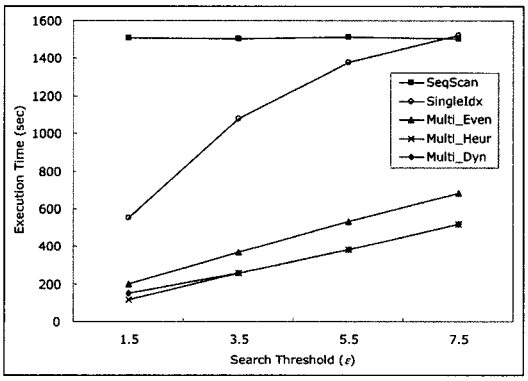
(a) 질의 집합 Q_{even}



(b) 질의 집합 Q_1

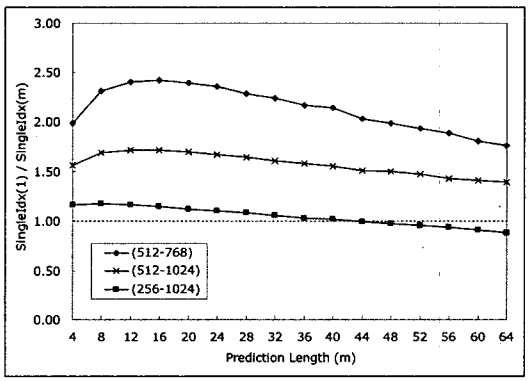


(c) 질의 집합 Q_2

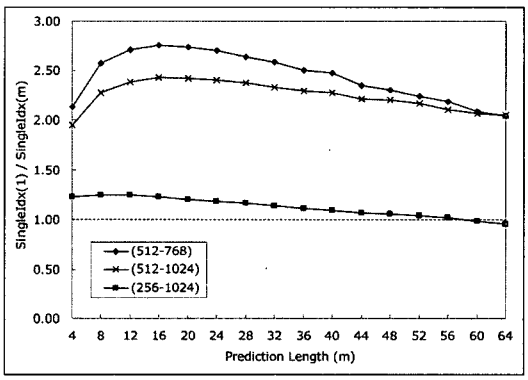


(d) 질의 집합 Q_4

그림 15 두번째 실험 결과



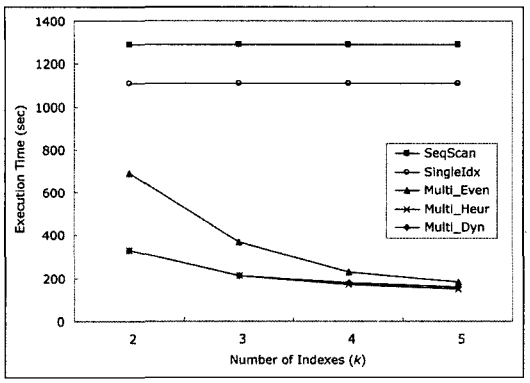
(a) 질의 집합 Q_{even}



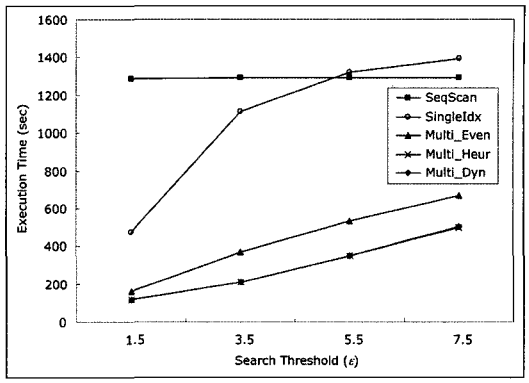
(b) 질의 집합 Q_1

그림 16 세번째 실험 결과

실험을 수행하였다. 검색 한계 ε은 첫번째 실험과 같이 3.5로, 인덱스의 개수 k는 두번째 실험과 같이 3으로 설정하였다. 그림 17은 질의 시퀀스 개수 변화에 따른 네 번째 실험 결과를 나타내며, 그래프의 세로 축은 로그 스케일(log scale)로 표시되었음을 유의하기 바란다. 그림 17의 실험 결과를 보면, 질의 시퀀스의 개수가 많아질수록 각 알고리즘 간의 성능 차이가 크게 커짐을 알 수 있다. 이는 질의 시퀀스 개수가 많아질수록 인덱스를 통한 필터링 효과가 커지기 때문이다. 즉, 다중 인덱스를 사용하는 알고리즘이 그렇지 않은 알고리즘들에 비하여 인덱스 검색 단계에서 보다 많은 질의 시퀀스를 필터링할 수 있기 때문이다. 이와 같은 실험 결과로 볼 때, 제안된 알고리즘은 대량의 질의 시퀀스 환경에서 보다 적합한 알고리즘이라 할 수 있다.



(a) 인덱스 개수(k)에 따른 실험 결과



(b) 검색 한계(ε)에 따른 실험 결과

그림 18 다섯번째 실험 결과

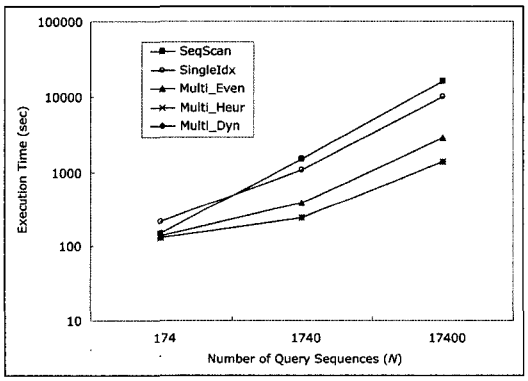


그림 17 네번째 실험 결과

다섯번째 실험에서는 실제 주가 데이타를 사용하여 본 논문에서 제안한 알고리즘의 우수성을 확인하였다. 실험에서 사용된 데이타는 1926년에서 1993년까지 Standard and Poor's 500 지수의 종가(closing price)로서, 총 17,610개의 요소 값으로 구성되어 있다.[6] 이러한 실제 주가 데이타를 스트리밍 시계열로 사용하였고, 질의 시퀀스 생성, 인덱스 구성 등은 앞에서 설명한 합성 데이타와 동일하게 수행하였다. 질의 집합으로는 Q_{even}을 사용하였으며, 인덱스의 개수(k)와 검색 한계(ε)를 변경하면서 실험하였다. 그림 18은 다섯번째 실험의 결과를 보인 것이다. 그림 18(a)는 검색 한계를 3.5로 설정하고 인덱스 개수를 2, 3, 4, 5로 변경하면서 실험한 결과이고, 그림 18(b)는 인덱스 개수를 3으로 설정하고 검색 한계를 1.5, 3.5, 5.5, 7.5로 변경하면서 실험한 결과이다. 그림 18의 실험 결과는 합성 데이타에 대한 실험 결과와 매우 유사하게 나타났다. 그림 18(a)의 결과는 그림 14(a)와 매우 유사하며, 제안한 *Multi_Heur* 알고리즘이 *SeqScan* 및 *SingleIdx* 등의 다른 알고리즘에 비해 우수한 결과를 보임을 확인할 수 있다. 또한, 그림 18(b)의 결과는 그림 15(a)와 매우 유사하며, 마찬가지로 *Multi_Heur* 알고리즘이 다른 알고리즘들에 비해서 우수한 성능을 보이고 있다. 결과적으로, 실제 주식 데이타를 사용한 실험에서도 합성 데이타를 사용한 경우와 동일하게, 본 논문에서 제안한 *Multi_Heur* 알고리즘이 다른 방법에 비해 성능을 크게 향상시킴을 확인할 수 있다.

8. 결 론

본 논문에서는 정규화 변환을 지원하는 스트리밍 서브시퀀스 매칭 문제를 해결하기 위한 효율적인 알고리즘을 제안하였다. 기존에는 스트리밍 시계열을 아무런 변환 없이 비교하였으나, 본 논문에서는 정규화 변환된 스트리밍 시계열을 비교한다. 정규화 변환은 절대적인 값은 달라도 유사한 변동 경향을 가지는 시계열 데이타를 찾기 위하여 유용하다[4]. 본 논문의 공헌은 다음과 같다. 기존의 정규화 변환을 지원하는 서브시퀀스 매칭 알고리즘[4]에서 제시한 정리를 이용하여 정규화 변환을 지원하는 스트리밍 서브시퀀스 매칭 문제를 풀기 위한 간단한 알고리즘을 제안하였다. 검색 성능을 향상시키기 위하여 간단한 알고리즘을 k 개의 인덱스를 이용하는 알고리즘으로 확장하였다. 주어진 k에 대하여, 확장된 알고리즘의 검색 성능을 최대화하기 위해 k 개의 최적의 윈도우 길이를 선택하기 위한 알고리즘을 제시하였다. 스트리밍 시계열의 연속성 개념[8]에 기반하여, 현재 시점 t_0에서 미래 시점 ($t_0 + m - 1$)까지의 검색 결과를 한번의 인덱스 검색으로 찾을 수 있도록 재차 확장

6) 실제 주가 데이타는 http://www.cs.ucr.edu/~eamonn/TSDMA/index.html에서 구할 수 있다.

한 알고리즘을 제안하였다. 일련의 실험을 통하여 본 논문에서 제안된 알고리즘들 간의 성능을 비교하고, k 및 m 값의 변화에 따라 제안된 알고리즘들의 검색 성능 변화를 보였다. 실험 결과, 제안된 알고리즘은 순차 검색 알고리즘에 비하여 최대 13.2 배까지 성능이 향상되었으며, 인덱스의 개수 k가 증가함에 따라 검색 성능도 함께 증가하였다.

참 고 문 헌

[1] R. Agrawal, C. Faloutsos, and A. Swami, "Efficient Similarity Search in Sequence Databases," In *Proc. the 4th Int'l Conf. on Foundations of Data Organization and Algorithms(FODO)*, Chicago, Illinois, pp. 69-84, Oct. 1993.

[2] 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, Minneapolis, Minnesota, pp. 419-429, May 1994.

[3] J. Han and M. Kamber, *Data Mining: Concepts and Techniques*, 2nd Ed., Morgan Kaufman, 2005.

[4] W.-K. Loh, S.-W., Kim, and K.-Y. Whang, "A Subsequence Matching Algorithm that Supports Normalization Transform in Time-Series Databases," *Data Mining and Knowledge Discovery*, Vol. 9, No. 1, pp. 5-28, July 2004.

[5] Y.-S. Moon, K.-Y., Whang, and W.-K. Loh, "Duality-Based Subsequence Matching in Time-Series Databases," In *Proc. the 17th Int'l Conf. on Data Engineering* (ICDE), IEEE, Heidelberg, Germany, pp. 263-272, April 2001.

[6] B.-K. Yi, H. V. Jagadish, and C. Faloutsos, "Efficient Retrieval of Similar Time Sequences Under Time Warping," In *Proc. the 14th Int'l Conf. on Data Engineering(ICDE)*, IEEE, Orlando, Florida, pp. 201-208, Feb. 1998.

[7] L. Gao and X. S. Wang, "Continually Evaluating Similarity-Based Pattern Queries on a Streaming Time Series," In *Proc. Int'l Conf. on Management of Data*, ACM SIGMOD, Madison, Wisconsin, pp. 370-381, June 2002.

[8] L. Gao, Z. Yao, and X. S. Wang, "Evaluating Continuous Nearest Neighbor Queries for Streaming Time Series via Pre-Fetching," In *Proc. ACM Int'l Conf. on Information and Knowledge Management (CIKM)*, McLean, Virginia, pp. 485-492, Nov. 2002.

[9] H. Wu, B. Salzberg, and D. Zhang, "Online Event-driven Subsequence Matching Over Financial Data Streams," In *Proc. of Int'l Conf. on Management of Data*, ACM SIGMOD, Paris, France, pp. 23-34, June 2004.

[10] B. Babcock, S. Babu, M. Datar, R. Motwani, and J. Widom, "Models and Issues in Data Stream Systems," In *Proc. ACM SIGACT-SIGMOD-SIGART Symp. on Principles of Database Systems (PODS)*, Madison, Wisconsin, pp. 1-16, June 2002.

[11] 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 Practices of Constraint Programming*, Cassis, France, pp. 137-153, Sept. 1995.

[12] E. J. Keogh, "A Decade of Progress in Indexing and Mining Large Time Series Databases," In *Proc. Int'l Conf. on Very Large Data Bases (VLDB)*, Tutorial, Seoul, Korea, pp. 1268, Sept. 2006.

[13] A. J. Frost, R. R. Prechter, and C. J. Collins, *Elliott Wave Principle: Key to Market Behavior*, John Wiley & Sons, 2001.

[14] N. Beckmann, H.-P. Kriegel, R. Schneider, and B. Seeger, "The R*-tree: An Efficient and Robust Access Method for Points and Rectangles," In *Proc. Int'l Conf. on Management of Data*, ACM SIGMOD, Atlantic City, New Jersey, pp. 322-331, May 1990.

[15] S. Berchtold, C. Bohm, and H.-P. Kriegel, "The Pyramid-Technique: Towards Breaking the Curse of Dimensionality," In *Proc. Int'l Conf. on Management of Data*, ACM SIGMOD, Seattle, Washington, pp. 142-153, June 1998.

[16] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, *Numerical Recipes in C: The Art of Scientific Computing*, Cambridge University Press, 2nd Ed., 1992.

[17] S.-H. Lim, H.-J. Park, and S.-W. Kim, "Using Multiple Indexes for Efficient Subsequence Matching in Time-Series Databases," In *Proc. Int'l Conf. on Database Systems for Advanced Applications (DASFAA)*, pp. 65-79, Singapore, Apr. 2006.

[18] W. R. Stevens and S. A. Rago, *Advanced Programming in the UNIX Environment*, 2nd Ed., Addison-Wesley, 2005.

[19] E. J. Keogh, L. Wei, X. Xi, S.-H. Lee, and M. Vlachos, "LB_Keogh Supports Exact Indexing of Shapes under Rotation Invariance with Arbitrary Representations and Distance Measures," In *Proc. Int'l Conf. on Very Large Data Bases (VLDB)*, Seoul, Korea, pp. 882-893, Sept. 2006.

노 웅 기

1991년 2월 한국과학기술원 전산학과 학사. 1993년 2월 한국과학기술원 전산학과 석사(멀티미디어 전공). 2001년 2월 한국과학기술원 전산학과 박사(데이타마이닝 전공). 2001년 2월~2003년 9월 ㈜티맥스소프트 책임연구원(미들웨어 개발). 2003년 10월~2005년 3월 ㈜티맥스데이타 수석연구원(DBMS 개발). 2005년 4월~2006년 5월 한국과학기술원 전산학과 초빙교수. 2006년 6월~현재 미국 University of

Minnesota 방문연구원. 관심분야는 데이타 마이닝/데이타 웨어하우징, 정보 검색, 멀티미디어 데이타베이스, 멀티미디어 내용기반 검색, 데이타베이스 시스템, 임베디드 데이타베이스

문 양 세
정보과학회논문지 : 데이타베이스
제 33 권 제 1 호 참조

김 영 국
1985년 2월 서울대학교 계산통계학과 학사. 1987년 2월 서울대학교 계산통계학과 석사. 1995년 5월 버지니아대 컴퓨터과학과 박사. 1995년 3월~1996년 2월 핀란드 VTT, 노르웨이 SINTEF DELAB 방문연구원. 1996년 3월~현재 충남대학교 전기정보통신공학부 교수. 2002년 8월~2003년 7월 UC Davis 방문교수. 관심분야는 실시간데이타베이스, 모바일정보시스템, 전자상거래시스템