시계열 스트림 데이터 상에서 핸드헬드 디바이스를 위한 효율적인 스트림 시퀀스 매칭 알고리즘

정회원 문 양 세*, 노 웅 기**

Efficient Stream Sequence Matching Algorithms for Handheld Devices over Time-Series Stream Data

Yang-Sae Moon*, Woong-Kee Loh** ***Regular Members***

요 약

핸드헬드 디바이스의 경우, 반복 작업에 대한 CPU 연산 최소화가 성능에 중요한 요소이다. 본 논문에서는 주식 데이터, 네트워크 트래픽, 센서 데이터 등의 시계열 스트림 데이터 상에서 유사 시퀀스를 효율적으로 찾아내는 핸드헬드 디바이스용 알고리즘을 제시한다. 이를 위하여, 우선 시계열 스트림 데이터 상에서 유사 시퀀스를 찾아내는 문제를 **스트림 시퀀스 매칭**(*stream sequence matching*)으로 정형적으로 정의한다. 다음으로, 기존의 서브시퀀스 매칭에서 사용했던 윈도우 구성법을 적용하여, 스트림 시퀀스 매칭을 효율적으로 처리하는 윈도우 기반 접근법을 제안한다. 그리고 이러한 윈도우 기반 접근법을 가능하게 하는 **윈도우** MBR(*window MBR*) 개념을 제시하고, 이 개념을 사용하면 스트림 시퀀스 매칭을 정확하게 수행할 수 있음을 증명한다. 또한, 윈도우 기반 접근법에 기반한 두 가지 스트림 시퀀스 매칭 알고리즘을 제안한다. 마지막으로, 분석과 실험을 통해 제안한 알고리즘이 단순 접근법에 비해 CPU 연산을 크게 줄이고 성능을 향상시킴을 보인다. 이 같은 결과를 볼 때, 제안한 방법은 CPU 연산 능력이 부족한 핸드헬드 디바이스의 내장형 알고리즘으로 매우 적합하다고 사료된다.

Key Words : Stream data, Time-series data, Stream sequence matching, Handheld devices

ABSTRACT

For the handheld devices, minimizing repetitive CPU operations such as multiplications is a major factor for their performances. In this paper, we propose efficient algorithms for finding similar sequences from streaming time-series data such as stock prices, network traffic data, and sensor network data. First, we formally define the problem of similar subsequence matching from streaming time-series data, which is called the stream sequence matching in this paper. Second, based on the window construction mechanism adopted by the previous subsequence matching algorithms, we present an efficient window-based approach that minimizes CPU operations required for stream sequence matching. Third, we propose a notion of window MBR and present two stream sequence matching algorithms based on the notion. Fourth, we formally prove correctness of the proposed algorithms. Finally, through a series of analyses and experiments, we show that our algorithms significantly outperform the naive algorithm. We believe that our window-based algorithms are excellent choices for embedded stream sequence matching in handheld devices.

※ 본 연구는 첨단정보기술연구센터를 통하여 과학기술부/한국과학재단의 지원을 받았음.
* 강원대학교 컴퓨터과학과 (ysmoon@kangwon.ac.kr)
** 한국과학기술원 전산학과/첨단정보기술연구센터 (woong@mozart.kaist.ac.kr)
논문번호 : KICS2005-07-001, 접수일자 : 2005년 7월 21일, 최종논문접수일자 : 2006년 7월 19일

I. 서 론

시계열 데이터란 각 시간별로 측정한 실수 값의 시퀀스로, 그 예로는 주식 데이터, 환율 변동 데이터, 날씨 데이터 등이 있다[4,9]. 현재까지는 이러한 시계열 데이터가 일반적으로 데이터베이스에 저장된다고 가정하고 많은 연구가 진행되었다[1,4,9,10]. 이와 같은 시계열 데이터베이스에 저장된 데이터를 **데이터 시퀀스**라 하고, 사용자에 의해 주어지는 시퀀스를 **질의 시퀀스**라 부른다. 그리고 주어진 질의 시퀀스와 유사한 데이터 시퀀스를 검색하는 방법을 **유사 시퀀스 매칭**이라 한다[4,9]. 이러한 유사 시퀀스 매칭은 유사한 가격 동향을 보이는 주식 종목의 발굴, 판매 경향이 유사한 상품의 식별, 네트워크 및 센서 데이터에서 유사 시퀀스 혹은 이상 시퀀스의 발견 등에서 널리 활용되고 있다[1,4,9,10].

그런데, 최근에는 대용량으로 끊임없이 발생하는 데이터를 효율적으로 처리하기 위한 스트림 데이터에 대한 연구가 활발히 진행되고 있다[2,6,12]. 스트림 데이터의 예로는 주식 종목의 실시간 가격 동향, 네트워크에서 발생하는 트래픽 데이터, 온도 및 기압 등의 측정에서 발생하는 센서 데이터 등이 있다. 이러한 스트림 데이터는 그 양이 너무 많고 또한 발생하는 속도가 너무 빨라서 모든 내용을 디스크, 즉 데이터베이스에 저장할 수 없으며 실시간으로 처리해야 하는 특징을 가진다[2]. 따라서 데이터베이스에 저장된 시계열 데이터가 아닌 끊임없이 발생하는 스트림 데이터에 대한 유사 시퀀스 매칭을 실시간 처리하는 연구가 필요하다.

본 논문에서는 시계열 데이터 형태로 발생하는 스트림 데이터(간략히 **스트림**이라 한다)를 대상으로 유사 시퀀스를 찾아내는 문제인 **스트림 시퀀스 매칭**(*stream sequence matching*)을 효율적으로 수행하는 알고리즘을 제시한다. 지금까지의 스트림 시퀀스 매칭은 일반적인 데이터베이스 환경에서 대량의 질의 시퀀스를 처리하는데 연구의 초점이 맞추어져 있었을 뿐[6,7], 핸드헬드 디바이스에 대한 연구는 이루어지지 않았다. 반면에, CPU 연산 능력이 부족한 핸드헬드 디바이스[5,8]의 경우, 지속적으로 발생하는 스트림을 모니터링 하면서 스트림 시퀀스 매칭을 수행하기 위해서 CPU 연산의 최소화가 중요한 요소이다. 그 이유는 유사 시퀀스를 찾는 일반적인 방법이 주어진 질의 시퀀스와 스트림 내의 시퀀스 사이의 거리 계산을 통해 이뤄지고, 이러한 거리 계산은 많은 CPU 연산을 필요로 하기 때문이다. 제안하는 스트림 시퀀스 매칭 알고리즘은 시퀀스 매칭 과정에서 필요로 하는 CPU 연산을 최소화함으로써, 휴대형 디바이스에서의 시퀀스 분석이나, 패킷 분석을 수행하는 네트워크 모니터링 장비[5]에 활용될 수 있다.

본 논문에서는 스트림 시퀀스 매칭을 위하여, 윈도우 기반 접근법을 제시한다. 윈도우 기반 접근법이란 시계열 서브시퀀스 매칭 연구[4,9]에서 활용되었던 "긴 시퀀스를 윈도우로 나누어 저장하고 비교하는 방법"을 스트림 시퀀스 매칭에 활용한 것이다. 본 논문에서는 기존 서브시퀀스 매칭의 이론을 스트림 시퀀스 매칭에 적용하기 위하여, 여러 윈도우들을 하나의 MBR(minimum bounding rectangle)에 포함시키는 **윈도우 MBR**(*window MBR*) 개념을 제시한다. 그리고, 이러한 윈도우 MBR에 기반하여 스트림 시퀀스 매칭을 정확하게 수행하는 이론적 근거를 정리로서 제시하고 증명한다. 다음으로, 윈도우 기반 접근법을 사용하여 스트림 시퀀스 매칭을 수행하는 두 가지 알고리즘을 제안한다. 마지막으로, 분석과 실험을 통해서 CPU 연산 및 실제 수행 시간 측면에서 제안한 방법의 우수성을 입증한다. 이와 같은 이론적 정확성과 성능의 우수성을 볼 때, 제안한 알고리즘은 CPU 연산 능력이 떨어지는 핸드헬드 디바이스에서 실시간 스트림 시퀀스 매칭을 효율적으로 수행할 수 있는 우수한 프레임워크를 제공한다.

본 논문의 구성은 다음과 같다. 제2장에서는 스트림 시퀀스 매칭 문제를 정의하고, 기존 연구를 설명한다. 제3장에서는 스트림 시퀀스 매칭을 수행할 수 있는 단순 접근법을 제시한다. 제4장에서는 제안하는 윈도우 기법 접근법을 설명한다. 제5장에서는 성능평가를 통해 제안한 방법의 우수성을 보이고, 마지막으로 제6장에서 결론을 맺는다.

II. 문제 정의 및 관련 연구

본 장에서는 우선 스트림 시퀀스 매칭 문제를 정형적으로 정의한다. 이를 위하여, 본 논문에서 사용하는 주요 표기법을 정리하면 다음과 같다. 먼저, 무한하게 발생하는 스트림을 S로 표현하고, 시점 i에 발생한 실수형의 스트림 엔트리를 $S[i]$라 표현하며, 시점 i에서 j까지 발생한 스트림 엔트리들의 집합을 S의 i에서 j까지 엔트리로 구성된 **스트림 시퀀스**라 하고 이를 $S[i:j]$라 표현한다. 이러한 정의 및 표기법을 사용하여 스트림 시퀀스 매칭을 다음과 같이 정형적으로 정의한다.

정의 1: 스트림 *S*에 대한 **스트림 시퀀스 매칭**이란, 스트림 *S*에 대해서, 질의 시퀀스 *Q*와의 거리가 허용치 ε이하인 스트림 시퀀스 *S*[*i*:*i*+*Len*(*Q*)-1]을 찾는 문제이다. 즉, 식 (1)이 성립하는 스트림 시퀀스 *S*[*i*:*i*+*Len*(*Q*)-1]을 찾는 문제이다.

$$D(Q, S[i:i+Len(Q)-1]) \le \epsilon \qquad (1)$$

여기에서, *Len*(*X*)는 시퀀스 *X*의 길이를 나타내며, *D*(*X*,*Y*)는 두 시퀀스 *X*(={*X*[1], *X*[2], ..., *X*[*n*]})와 *Y*(={*Y*[1], *Y*[2], ..., *Y*[*n*]})에 대한 유클리디안 거리 $\left(= \sqrt{\sum_{i=1}^{n}(X[i]-Y[i])^2}\right)$이다. □

그리고, 본 논문에서는 스트림 *S*의 크기 ω인 **슬라이딩 윈도우**를 *S*의 모든 엔트리를 시작 위치로 하는 크기 ω의 스트림 시퀀스 *S*[*a*:*a*+ω-1]로, 크기 ω인 **디스조인트 윈도우**를 *S*의 엔트리들을 대상으로 한번에 ω만큼씩 이동하면서 구성한 크기 ω의 스트림 시퀀스 *S*[(*b*-1)ω+1:*b*ω]로 각각 정의한다[9].

유사 시퀀스 매칭에 대해서는 시계열 데이터베이스를 기반으로 서브시퀀스 매칭 연구가 수행되었다[4,9,10]. 이들 방법에서는 데이터베이스에 저장된 데이터 시퀀스를 윈도우로 나누어 저차원 변환한 후 다차원 색인을 구성한다. 그런 다음, 사용자에 의해 주어진 질의 시퀀스 역시 윈도우로 나누어 저차원 변환하고, 허용치와 함께 다차원 색인을 검색하여 후보 시퀀스를 구한 후, 실제 데이터베이스를 액세스하여 유사 시퀀스만을 찾는다. 그러나, 이들 방법은 시계열 데이터가 데이터베이스에 저장되어 있다고 가정하였기 때문에 무한하게 발생하는 실시간 스트림 분석에는 적합하지 않다. 그리고, 기존 방법에서 사용한 저차원 변환[3,10]은 많은 CPU 연산을 추가적으로 필요로 하고, 이에 따라 CPU 연산 능력이 약한 핸드헬드 디바이스에 적용이 어려운 문제점이 있다. 또한, 기존 방법은 다차원 색인 구성을 위한 별도의 저장 공간을 필요로 하기 때문에, 작은 저장 공간만을 가진 핸드헬드 디바이스에는 적합하지 않다. 따라서, 본 논문에서는 핸드헬드 디바이스에 적합하도록 CPU 연산을 줄임과 동시에 추가적인 색인 공간 영역이 필요로 하지 않는 효율적인 스트림 시퀀스 매칭 방법을 제시한다.

Ⅲ. 스트림 시퀀스 매칭의 단순 접근법

스트림 시퀀스 매칭을 위해서 가장 쉽게 생각할 수 있는 단순 접근법은 스트림 내의 모든 가능한

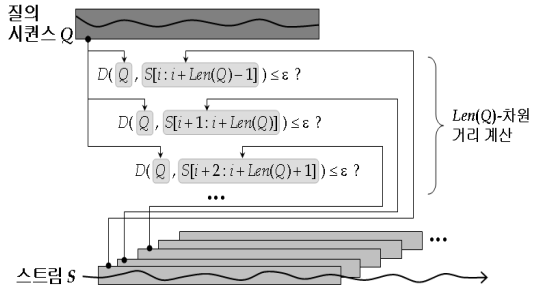


그림 1. 스트림 시퀀스 매칭의 단순 접근법

스트림 시퀀스를 질의 시퀀스와 비교하는 것이다. 즉, 모든 시점 *i*에 대한 스트림 시퀀스인 *S*[*i*:*i*+*Len*(*Q*)-1]과 질의 시퀀스 *Q*와의 거리를 계산하여, 그 거리가 사용자가 제시한 ε이하인지 검사하는 것이다. 그림 1은 이러한 단순 접근법의 유사 시퀀스 매칭 과정을 나타낸다. 그림을 보면, 단순 접근법은 주어진 질의 시퀀스 *Q*에 대해서 스트림 *S*의 모든 스트림 시퀀스에 대해서 각각 거리를 계산하여 그 거리가 ε이내에 있는지를 검사한다.

그런데, 이러한 단순 접근법은 너무 많은 거리 연산을 필요로 하는 문제점이 있다. 즉, 단순 접근법은 스트림 *S*에서 새로운 엔트리 하나가 발생할 때마다 *Len*(*Q*)-차원 거리 계산을 필요로 한다. 그리고, 이러한 *Len*(*Q*)-차원 거리 계산은 유클리디안 거리의 정의에 따라 *Len*(*Q*)번의 실수 곱셈 연산을 필요로 한다(한 번의 제곱근 연산은 곱셈에 비해 그 수가 적으므로 분석에서 제외하고, *Len*(*Q*)번의 뺄셈 연산은 *Len*(*Q*)번의 곱셈 연산에 포함되는 것으로 간주한다.). 만일, 스트림의 길이가 *L*이라 가정하면, 단순 접근법의 경우, (*L*-*Len*(*Q*)+1) · *Len*(*Q*)번의 곱셈 연산을 필요로 하게 된다(단순 접근법의 원리에 의해 쉽게 계산되므로 자세한 내용은 생략한다.).

Ⅳ. 스트림 시퀀스 매칭의 윈도우 접근법

본 장에서는 시계열 데이터베이스에서의 대표적 서브시퀀스 매칭 방법인 Faloutsos 등[4]의 연구(간략히 *FRM*이라 한다)와 DualMatch[9] 연구의 윈도우 구성법 각각을 스트림 시퀀스 매칭에 적용한 방법을 제시한다. 제4.1절에서는 FRM의 윈도우 구성법을 스트림 시퀀스 매칭에 적용한 *FRM-SSM*(FRM-based Stream Sequence Matching)을 제안하고, 제4.2절에서는 DualMatch의 윈도우 구성법을 적용한 DM-SSM (DualMatch-based Stream Sequence Matching)을 제안한다.

4.1 FRM-SSM: FRM-based Approach

FRM에서는 데이터 시퀀스 X를 크기 ω의 슬라이딩 윈도우 $X[a:a+\omega-1]$로 나누고, 질의 시퀀스 Q를 같은 크기의 디스조인트 윈도우 $Q[(b-1)\omega+1:b\omega]$로 나누는 윈도우 구성법을 사용한다. 그리고 다음 보조정리 1을 사용하여 서브시퀀스 매칭을 수행한다.

보조정리 1[4]: 질의 시퀀스 Q와 데이터 시퀀스 X의 서브시퀀스 $X[i:i+Len(Q)-1]$의 거리가 ε이하이면, 적어도 하나 이상의 Q의 디스조인트 윈도우 $Q[(b-1)\omega+1:b\omega]$와 $X[i:i+Len(Q)-1]$의 슬라이딩 윈도우 $X[i+(b-1)\omega:i+b\omega-1]$의 거리가 $\varepsilon/\sqrt{p}$ 이하이다. 즉, 다음 식 (2)가 성립한다. 여기에서, $p = \lfloor Len(Q)/\omega \rfloor$, $1 \le b \le p$, $1 \le i \le Len(X) - \omega + 1$이다.

$$D(Q, X[i:i+Len(Q)-1]) \le \varepsilon \Rightarrow \bigvee_{b=1}^{p} \left(D(Q[(b-1)\cdot\omega+1:b\cdot\omega], X[i+(b-1)\cdot\omega:i+b\cdot\omega-1]) \le \varepsilon/\sqrt{p} \right) \quad (2)$$

보조정리 1에 따라서, FRM에서는 Q의 디스조인트 윈도우 $Q[(b-1)\omega+1:b\omega]$와 X의 슬라이딩 윈도우 $X[i+(b-1)\omega:i+b\omega-1]$이 $\varepsilon/\sqrt{p}$ 거리 이하일 때, 즉 식 (2)의 필요조건이 만족할 때, X의 서브시퀀스 $X[i:i+Len(Q)-1]$를 Q와 유사할 가능성이 높은 후보 시퀀스로 삼는다.

FRM에서 사용한 보조정리 1을 FRM-SSM에 적용하기 위하여, 다음과 같이 연속적인 슬라이딩 윈도우들을 포함하는 윈도우 MBR을 정의한다.

정의 2: 크기 ω인 연속적인 슬라이딩 윈도우 $W[i:i+\omega-1]$, $W[i+1:i+\omega]$, ..., $W[j:j+\omega-1]$들의 집합을 $W\{i:j\}$라 했을 때, 이들 윈도우를 포함하는 ω-차원의 MBR을 ***W{i:j}의 윈도우 MBR***이라 정의하고, 이를 $mbr(W\{i:j\})$로 표기한다. □

이와 같이 연속된 윈도우들의 집합 $W\{i:j\}$의 윈도우 MBR을 정의하는 이유는, 보조정리 1을 FRM-SSM에 그대로 적용하면 단순 접근법 못지않게 많은 곱셈 연산이 필요하기 때문이다. 즉, 지속적으로 발생하는 스트림 S에 있어서, 모든 시점에 대한 슬라이딩 윈도우와 질의 시퀀스의 디스조인트 윈도우에 대한 ω-차원 거리 계산을 수행해야 하기 때문이다.

FRM-SSM에서는 ω-차원의 거리 계산을 줄이기 위하여, 보조정리 1에 윈도우 MBR 개념을 적용한 다음 정리 1을 사용한다.

정리 1: 스트림 S에서 시점 i에 대한 스트림 시퀀스 $S[i:i+Len(Q)-1]$와 질의 시퀀스 Q와의 거리가 ε이하이면, 적어도 하나 이상의 Q의 디스조인트 윈도우 $Q[(b-1)\omega+1:b\omega]$와 시점 x에서 y까지를 시작 위치로 하는 S의 슬라이딩 윈도우로 구성된 윈도우 MBR인 $mbr(S\{x:y\})$의 거리가 $\varepsilon/\sqrt{p}$ 이하이다. 즉, 다음 식 (3)이 성립한다. 여기에서, $x \le i+(b-1)\omega \le y$, $1 \le b \le p$, $p = \lfloor Len(Q)/\omega \rfloor$이다.

$$D(Q, S[i:i+Len(Q)-1]) \le \varepsilon \Rightarrow \bigvee_{b=1}^{p} \left(D(Q[(b-1)\cdot\omega+1:b\cdot\omega], mbr(S\{x:y\})) \le \varepsilon/\sqrt{p} \right) \quad (3)$$

증명: 보조정리 1과 윈도우 MBR 정의로 쉽게 증명할 수 있으므로 자세한 과정은 생략한다. □

정리 1에 의하여, FRM-SSM에서는 스트림 S의 여러 슬라이딩 윈도우들을 대상으로 MBR을 구성한다. 그런 다음, Q의 디스조인트 윈도우인 $Q[(b-1)\omega+1:b\omega]$와 S의 슬라이딩 윈도우 $S[i+(b-1)\omega:i+b\omega-1]$을 포함하는 $mbr(S\{i:j\})$의 거리가 $\varepsilon/\sqrt{p}$ 이하이면, 스트림 시퀀스 $S[i:i+Len(Q)-1]$를 후보 시퀀스로 삼는다. 즉, 식 (3)의 필요조건이 만족할 때, 스트림 시퀀스 $S[i:i+Len(Q)-1]$을 후보로 삼는 방법을 사용한다. 그리고, 후보가 된 스트림 시퀀스에 대해서는 질의 시퀀스와의 실제 $Len(Q)$-차원 거리 계산을 통하여 유사 시퀀스인지 확인한다. 이와 같이, FRM-SSM은 정리 1에 의하여 모든 유사 스트림 시퀀스를 빠짐없이 찾아내므로 정확성을 보장한다.

그림 2는 FRM-SSM의 스트림 시퀀스 매칭 알고리즘을 나타낸다. 알고리즘의 입력 변수는 무한 스트림 S, 사용자에 의해 주어지는 질의 시퀀스 Q와 허용치 ε이다. 그리고 알고리즘의 출력은 주어진 질의 시퀀스와 ε거리 이내에 있는 스트림 시퀀스들이다. 알고리즘을 보면, 단계 (1)에서 (9)까지 무한 루프를 수행하면서, 단계 (2)에서는 스트림의 슬라이딩 윈도우들에 대한 윈도우 MBR을 구성하고, 단계 (3)에서는 이 MBR이 질의 시퀀스를 나눈 디스조인트 윈도우와 $\varepsilon/\sqrt{p}$ 거리에 있는지를 확인하여 후보

```
Procedure FRM-SSM (Stream S, Query Sequence Q, Tolerance ε)
(1)  Loop forever
(2)    Make mbr(S{x:y}) using entries from x to y+ω-1;
(3)    if D(Q[(b-1)·ω+1:b·ω], mbr(S{x:y})) ≤ ε/√p then
(4)      // Consider S[i:i+Len(Q)-1] as a candidate.
(5)      if D(Q, S[i:i+Len(Q)-1]) ≤ ε then
(6)        Return S[i:i+Len(Q)-1] as a similar stream sequence;
(7)      endif
(8)    endif
(9)  end loop
```

그림 2. FRM-SSM의 스트림 시퀀스 매칭 알고리즘

시퀀스들을 찾은 후, 단계 (5)와 (6)에서는 이들 후보를 대상으로 질의 시퀀스와의 실제 *Len*(*Q*)-차원 거리를 계산하여 거리가 ε이하인 스트림 시퀀스를 찾아 유사 시퀀스로 반환한다.

4.2 DM-SSM: DualMatch-based Approach

다음으로, DualMatch의 윈도우 구성법에 기반한 스트림 시퀀스 매칭 알고리즘을 제안한다. 윈도우 구성법에 있어서 DualMatch는 FRM의 이원적 접근법으로, 데이터 시퀀스 *X*를 크기 ω의 디스조인트 윈도우 *X*[(*a*-1)ω+1:*a*ω]로 나누고, 질의 시퀀스 *Q*를 같은 크기의 슬라이딩 윈도우 *Q*[*b*:*b*+ω-1]로 나누는 윈도우 구성법을 사용한다. 그리고, 다음 보조정리 2을 사용하여 서브시퀀스 매칭을 수행한다.

보조정리 2[9]: 질의 시퀀스 *Q*와 데이터 시퀀스 *X*의 서브시퀀스 *X*[*i*:*i*+*Len*(*Q*)-1]의 거리가 ε이하이면, 적어도 하나 이상의 *Q*의 슬라이딩 윈도우 *Q*[*j*:*j*+ω-1]와 *X*[*i*:*i*+*Len*(*Q*)-1]에 포함된 디스조인트 윈도우 *X*[(*k*-1)ω+1:*k*ω]의 거리가 $\epsilon/\sqrt{p}$ 이하이다. 즉, 다음 식 (4)가 성립한다. 여기에서, $j=(k-1)\omega-i+2$, $p=\lfloor (Len(Q)+1)/\omega \rfloor -1$, $k=\lceil (i-1)/\omega \rceil +a$, $1 \le a \le p$ 이다.

$$D(Q, X[i:i+Len(Q)-1]) \le \varepsilon \Rightarrow \bigvee_{a=1}^{p} \left(D(Q[j:j+\omega-1], X[(k-1)\cdot\omega+1:k\cdot\omega]) \le \varepsilon/\sqrt{p} \right) \quad (4)$$

보조정리 2에 따라서, DualMatch에서는 *X*의 디스조인트 윈도우 *X*[(*k*-1)ω+1:*k*ω]와 *Q*의 슬라이딩 윈도우 *Q*[*j*:*j*+ω-1]의 거리가 $\epsilon/\sqrt{p}$ 이하일 때, 즉 수식 (4)의 필요조건이 만족할 때, 서브시퀀스 *X*[*i*:*i*+*Len*(*Q*)-1]이 *Q*와 유사할 가능성이 높은 후보 시퀀스로 삼는다.

DualMatch를 스트림 시퀀스 매칭에 직접 적용할 경우, 스트림을 나눈 윈도우들에 대해서는 MBR을 구성하지 않아도 되나, 질의 시퀀스를 나눈 윈도우들을 대상으로는 MBR을 구성하여야 한다. 그 이유는 스트림을 디스조인트 윈도우로 나누므로, FRM-SSM에 비하여 윈도우 개수를 크게(약 1/ω) 줄여 MBR을 구성하지 않아도 되는 반면에[9], 질의 시퀀스를 슬라이딩 윈도우로 나누므로, 질의 시퀀스에서 너무 많은 윈도우가 생성되는 문제점이 있기 때문이다. 따라서, FRM-SSM에서와 유사하게, DM-SSM에서는 질의 시퀀스를 나눈 이들 윈도우를 대상으로 윈도우 MBR을 구성한다. 즉, FRM-SSM에서와 마찬가지로 ω-차원 거리 계산을 줄이기 위해서, DM-SSM에서도 윈도우 MBR 개념을 도입한 다음 정리 2를 사용한다.

정리 2: 스트림 *S*에서 시점 *i*에 대한 스트림 시퀀스 *S*[*i*:*i*+*Len*(*Q*)-1]와 질의 시퀀스 *Q*와의 거리가 ε이하이면, 적어도 하나 이상의 *S*[*i*:*i*+*Len*(*Q*)-1]에 포함된 디스조인트 윈도우 *S*[(*k*-1)ω+1:*k*ω]와 *Q*의 슬라이딩 윈도우로 구성된 *mbr*(*Q*{1:*Len*(*Q*)-ω+1})의 거리가 $\epsilon/\sqrt{p}$ 이하이다. 즉, 다음 식 (5)가 성립한다. 여기에서, $p=\lfloor (Len(Q)+1)/\omega \rfloor -1$, $k=\lceil (i-1)/\omega \}$, $1 \le a \le p$이다.

$$D(Q, S[i:i+Len(Q)-1]) \le \varepsilon \Rightarrow \bigvee_{a=1}^{p} \left(D\left(mbr(Q\{1:Len(Q)-\omega+1\}), S[(k-1)\cdot\omega+1:k\cdot\omega]\right) \le \varepsilon/\sqrt{p} \right) \quad (5)$$

증명: 보조정리 2와 윈도우 MBR 정의로 쉽게 증명할 수 있으므로 자세한 과정은 생략한다. □

정리 2에 의하여, DM-SSM에서는 질의 시퀀스 *Q*의 모든 슬라이딩 윈도우들을 대상으로 윈도우 MBR을 구성한다. 그런 다음, 구성한 MBR과 스트림 *S*에서 구성되는 디스조인트 윈도우인 *S*[(*k*-1)ω+1:*k*ω]의 거리가 $\epsilon/\sqrt{p}$ 이하이면, 스트림 시퀀스 *S*[*i*:*i*+*Len*(*Q*)-1]를 후보 시퀀스로 삼는다. 즉, 수식 (5)의 필요조건이 만족할 때, *S*[*i*:*i*+*Len*(*Q*)-1]을 후보로 삼는 방법을 사용한다. 이와 같이, DM-SSM은 정리 2에 의하여 모든 유사 스트림 시퀀스를 빠짐없이 찾아내므로 정확성을 보장한다.

그림 3은 DM-SSM의 스트림 시퀀스 매칭 알고리즘을 나타낸다. 알고리즘의 입력 변수와 출력 변수는 FRM-SSM의 알고리즘과 동일하다. DM-SSM의 알고리즘을 보면, 윈도우 구성에 있어서 이원적 접근법을 사용하는 점과 스트림 *S*에 대해서 윈도우 MBR을 구성하는 대신에 질의 시퀀스 *Q*를 대상으로 윈도우 MBR을 구성하는 점을 제외하고는 FRM-SSM의 알고리즘과 동일하다.

```
Procedure DM-SSM (Stream S, Query Sequence Q, Tolerance ε)
(1)  Loop forever
(2)    Make mbr(Q{1:Len(Q)-ω+1}) using entries from 1 to Len(Q);
(3)    if D(mbr(Q{1:Len(Q)-ω+1}),S[(k-1)·ω+1:k·ω]) ≤ ε/√p then
(4)      // Consider S[i:i+Len(Q)-1] as a candidate.
(5)      if D(Q,S[i:i+Len(Q)-1]) ≤ ε then
(6)        Return S[i:i+Len(Q)-1] as a similar stream sequence;
(7)      endif
(8)    endif
(9)  end loop
```

그림 3. DM-SSM의 스트림 시퀀스 매칭 알고리즘

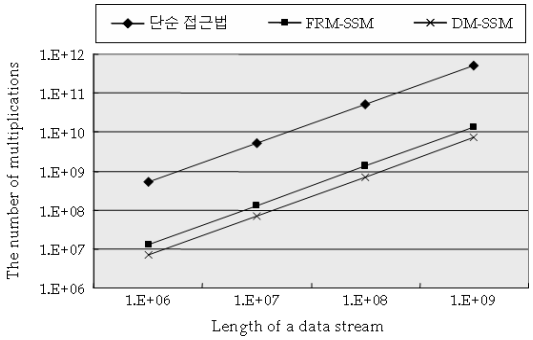


그림 4. 스트림 길이 변화에 따른 곱셈 횟수 비교

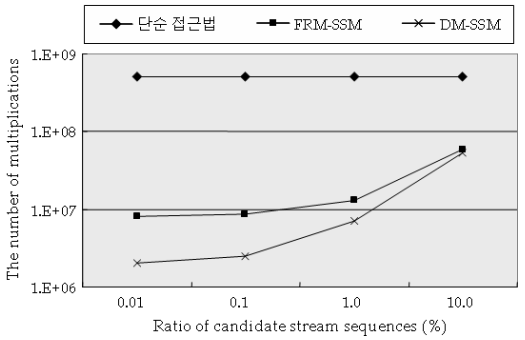


그림 5. 후보 개수 비율 변화에 따른 곱셈 횟수 비교

Ⅴ. 성능 평가

본 장에서는 단순 접근법, FRM-SSM, DM-SSM에 대한 성능 평가 결과를 설명한다. 제5.1절에서는 이들 방법에 대한 곱셈 횟수를 분석적 방법으로 평가하고, 제5.2절에서는 실험을 통해 이들 스트림 시퀀스 매칭 방법들의 실제 수행 시간을 평가한다.

5.1 분석 평가

본 절에서는 분석적 방법을 통해 세 가지 방법의 곱셈 횟수를 비교한다. 먼저, 단순 접근법의 경우, 제3장에서 설명한 바와 같이, 스트림 시퀀스의 길이가 L이라 하면, $(L\text{-}Len(Q)+1) \cdot Len(Q)$번의 곱셈 연산이 필요하다. 반면에, FRM-SSM 및 DM-SSM의 경우는 후보가 되는 시퀀스의 비율에 따라서 곱셈 연산의 횟수가 달라질 수 있다. 이러한 점을 고려하여, 후보가 되는 시퀀스의 비율을 전체의 α%라 하고, FRM-SSM과 DM-SSM의 곱셈 연산 횟수를 계산하면 다음과 같다(다음 수식은 FRM-SSM과 DM-SSM의 알고리즘에서 쉽게 구할 수 있으므로 자세한 계산 과정은 생략한다.).

- FRM-SSM에서 하나의 윈도우 MBR에 포함되는 스트림의 슬라이딩 윈도우 개수를 n_{mbr}이라 했을 때, FRM-SSM의 곱셈 연산의 총 회수는 $(2 \cdot L \cdot Len(Q))/n_{mbr}+(\alpha/100) \cdot L \cdot Len(Q)$이다.
- DM-SSM에서 필요한 곱셈 연산의 총 횟수는 $2 \cdot L+(\alpha/100) \cdot L \cdot Len(Q)$이다.

상기 곱셈 횟수의 계산식을 바탕으로, 스트림 길이, 후보 시퀀스 비율, 질의 시퀀스 길이를 각각 달리하면서 세 가지 분석 평가를 수행하였다.

분석 평가 1: 그림 4는 질의 시퀀스 길이를 512로, 후보 시퀀스의 비율을 전체 스트림 시퀀스의 1.0%로, FRM-SSM의 윈도우 크기를 256으로, DM-SSM의 윈도우 크기를 128로, FRM의 n_{mbr}을 128로 고정한 후, 스트림의 길이를 100만에서 10억개 엔트리까지로 증가시키면서 곱셈 횟수를 계산한 그래프이다. 그림을 보면, FM-SSM과 DM-SSM 모두 단순 접근법에 비해서 곱셈 횟수를 크게, 약 39배~72배까지 각각 줄인 것으로 나타났다.

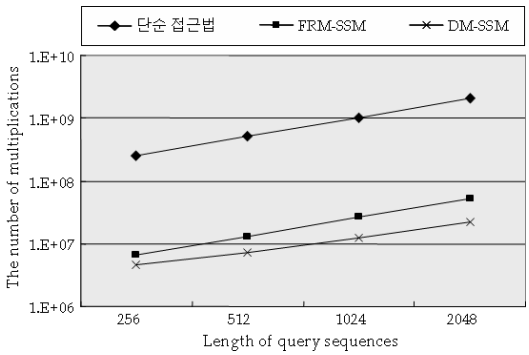


그림 6. 질의 시퀀스 길이에 따른 곱셈 연산 횟수 비교

분석 평가 2: 그림 5는 후보 개수 비율을 0.01%에서 10.0%까지 변화시키면서 곱셈 횟수를 계산한 그래프이다. 여기에서, 스트림 길이는 100만개 엔트리를 사용하였으며, 나머지 변수는 분석 평가 1과 동일한 값을 사용하였다. 그림에서 보면, FM-SSM과 DM-SSM은 여전히 단순 접근법에 비해서 곱셈 횟수를 크게 줄인 것으로 나타났다. 그런데, 후보 개수 비율이 증가할수록 FRM-SSM 및 DM-SSM과 단순 접근법의 차이가 줄어듦을 볼 수 있다. 이는 후보 개수 비율이 클수록 FRM-SSM과 DM-SSM에서 각 후보 시퀀스들을 대상으로 수행하는 실제 $Len(Q)$-차원 거리 계산이 많아지기 때문이다.

분석 평가 3: 그림 6은 질의 시퀀스의 길이를 256에서 2048까지 변화시키면서 곱셈 횟수를 계산한

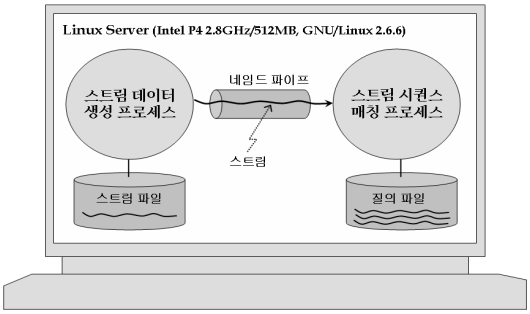


그림 7. 수행 시간 측정을 위한 실험 환경.

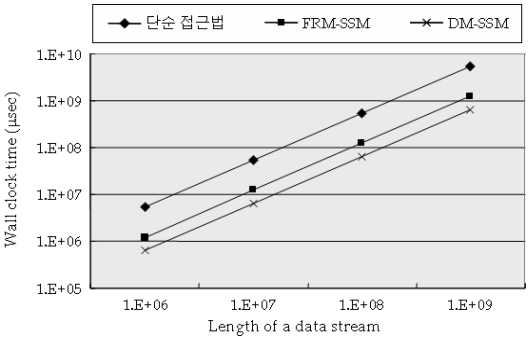


그림 8. 스트림의 길이 변화에 따른 수행 시간 비교.

그래프이다. 여기에서, 후보 개수 비율은 1.0%를 사용하였으며, 나머지 변수는 분석 평가 2와 동일한 값을 사용하였다. 그림을 보면, 앞서의 두 분석 평가와 마찬가지로 제안한 두 방법이 단순 접근법에 비해서 곱셈 횟수를 크게 줄인 것으로 나타났다.

5.2 실험 평가

본 절에서는 제안한 방법의 실용적 우수성을 입증하기 위하여, 실험을 통해 세 가지 방법의 실제 수행 시간을 비교하였다. 제5.2.1절에서는 실험을 수행한 환경 및 실험 데이터를 설명하고, 제5.2.2절에서는 실험 결과를 설명한다.

5.2.1 실험 환경 및 실험 데이터

스트림 환경을 모델링하기 위하여 Linux 운영체제 환경에서 그림 7과 같은 실험 환경을 구축하였다. 그림에서와 같이, Linux 서버에서 두 개의 프로세스가 네임드 파이프[11]를 사용하여 통신을 수행한다. 그림에서, 왼편의 스트림 데이터 생성 프로세스는 스트림 파일을 읽어서 지속적으로 스트림을 생성하고, 이를 통신 수단인 파이프를 통해서 스트림 시퀀스 매칭 프로세스로 전달한다. 오른편의 스트림 시퀀스 매칭 프로세스는 파이프로부터 스트림을 읽어 들여서, 주어진 질의 시퀀스와 유사한 스트림 시퀀스를 찾는 역할을 수행한다. 실험을 수행한 하드웨어 플랫폼은 Intel Pentium 4 2.80GHz CPU, 512MB RAM, 70.0GB 하드디스크를 장착한 PC이며, 소프트웨어 플랫폼은 GNU/Linux Version 2.6.6 운영체제이다.

실험에 사용한 스트림 파일은 합성 데이터(synthetic data)로서, 주식 데이터 및 환율 데이터 등을 모델링하는데 사용되는 랜덤 워크 데이터이다[4,9]. 스트림 파일의 생성은 시작 엔트리를 1.5로 하고, 각 엔트리에(−0.001, 0.001) 사이의 임의의 값 하나를 더하여 다음 엔트리를 구하는 방식을 사용하였다[4,9,10]. 100만개 엔트리의 스트림 파일을 먼저 생성한 후, 이를 10번 및 100번 반복하여 1000만개 및 1억개 엔트리의 스트림을 각각 생성하였다.

질의 시퀀스는 스트림 파일의 임의 위치를 시작 엔트리로 하는 $Len(Q)$의 스트림 시퀀스를 추출하여 사용하였으며, 노이즈 효과를 피하기 위하여 같은 길이를 갖는 10개의 다른 질의 시퀀스에 대해서 실험한 후 평균을 취한 값을 실험 결과로 하였다. 그런데, 실제 실험에서는 후보 개수가 질의의 종류 및 길이, 허용치 등에 의해 달라질 수 있어, 분석 평가에서 사용한 후보 개수 비율을 고정 값으로 사용할 수 없다. 따라서, 본 실험에서는 후보 개수 비율 대신에 선택률 값을 실험에 사용하였다. 여기에서, 선택률이란 모든 가능한 스트림 시퀀스 개수에 대한 질의 결과 얻은 유사 스트림 시퀀스 개수의 비율[4,9]로 정의된다.

5.2.2 실험 결과

실험은 분석 평가에서와 마찬가지로 세 가지 방법에 대해서, 스트림 길이, 선택률, 질의 시퀀스 길이를 각각 달리하면서 세 가지 평가를 수행하였다.

실험 평가 1: 그림 8은 분석 평가 1과 마찬가지로, 질의 시퀀스 길이를 512로, FRM-SSM의 윈도우 크기를 256로, DM-SSM의 윈도우 크기를 128로, FRM의 n_{mbr}을 128로 고정한 후, 스트림의 길이를 100만에서 10억개 엔트리까지로 증가시키면서 수행 시간을 측정한 그래프이다. 단, 실제 실험에서는 후보 시퀀스의 비율이 가변적이므로, 분석 평가 1에서 사용한 후보 비율 대신에 선택률을 0.0001로 사용하여 실험하였다. 그림을 보면, FM-SSM과 DM-SSM 모두 단순 접근법에 비해서 수행 시간을 각각 약 4.4배~8.6배까지 크게 줄인 것으로 나타났다. 분석 평가 1의 곱셈 횟수에 비해서 수행 시간의

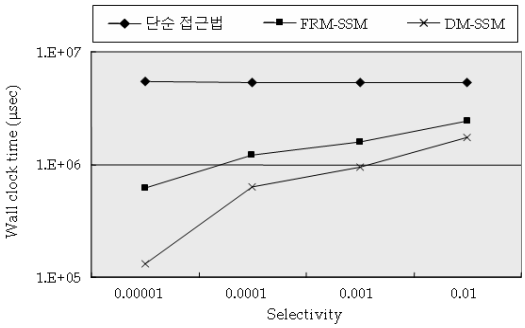


그림 9. 선택률 변화에 따른 수행 시간 비교

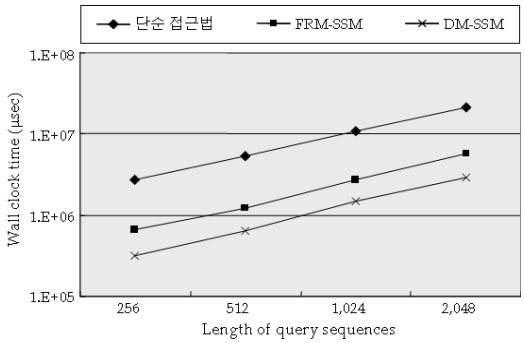


그림 10. 질의 시퀀스 길이 변화에 따른 수행 시간 비교

개선 효과가 작은 이유는 윈도우 관리, MBR의 구성, 통신 연산 등 곱셈 연산을 제외한 다른 부분의 오버헤드가 작용하기 때문이다.

실험 평가 2: 그림 9는 선택률을 0.00001에서 0.01까지 변화시키면서 수행 시간을 측정한 그래프이다. 후보 개수 비율 대신에 선택률을 사용한 것을 제외하고는 분석 평가 2와 동일한 환경을 사용하였다. 그림에서 보면, FM-SSM과 DM-SSM은 여전히 단순 접근법에 비해서 수행 시간을 크게 줄인 것으로 나타났다. 특히, 선택률이 낮은 경우(0.00001)에, FRM-SSM 및 DM-SSM은 단순 접근법에 비해 평균 8.8배 및 41.3배까지 각각 성능을 향상시킨 것으로 나타났다. 또한, 분석 평가 2에서와 마찬가지로, 선택률이 증가할수록 제안한 방법과 단순 접근법의 차이가 줄어듦을 볼 수 있다.

실험 평가 3: 그림 10은 분석 평가 3과 마찬가지로 질의 시퀀스의 길이를 256에서 2048까지 변화시키면서 수행 시간을 측정한 그래프이다. 여기에서, 선택률은 0.0001을 사용하였으며, 나머지 변수는 분석 평가 3과 동일한 값을 사용하였다. 그림을 보면, 분석 평가 3의 곱셈 횟수와 마찬가지로 수행 시간 역시 FRM-SSM 및 DM-SSM이 단순 접근법에 비해서 우수함을 알 수 있다.

VI. 결 론

최근 네트워크 트래픽, 센서 데이터, 주식 및 환율 데이터 등에서 스트림 데이터 처리가 중요한 이슈가 되고 있다. 이중에서 시계열 스트림 데이터의 분석은 주식 및 환율 데이터에서 유사 패턴의 발견, 네트워크 및 센서 데이터에서 유사 및 이상 시퀀스의 발견 등에 활용될 수 있다. 특히, CPU 연산 능력이 낮은 핸드헬드 디바이스에서도 이들 시계열 스트림 데이터에 대한 실시간 분석이 요구되고 있다. 본 논문에서는 핸드헬드 디바이스에 적합한 시계열 스트림 데이터에서의 유사 시퀀스 매칭 알고리즘을 제안하였다. 제안한 알고리즘은 유사 시퀀스 매칭에서 필요로 하는 CPU 연산을 최소화하여 수행 시간을 크게 줄이는 특징을 가진다.

본 논문의 공헌을 요약하면 다음과 같다. 우선, 시계열 스트림 데이터에 대한 유사 시퀀스 매칭을 스트림 시퀀스 매칭으로 정형적으로 정의하였다. 다음으로, 기존 서브시퀀스 매칭 연구에서 사용했던 윈도우 구성법을 적용하여, 스트림 시퀀스 매칭을 효율적으로 수행하는 윈도우 기반 접근법을 제안하였다. 그리고, 이러한 윈도우 기반 접근법을 가능하게 하기 위하여 윈도우 MBR 개념을 제시하고, 이 개념을 사용하면 스트림 시퀀스 매칭을 정확하게 수행할 수 있음을 증명하였다. 다음으로, 윈도우 기반 접근법에 기반한 두 가지의 스트림 시퀀스 매칭 알고리즘을 제안하였다. 또한, 분석과 실험을 통해 제안한 알고리즘이 단순 접근법에 비해 CPU 연산을 크게 줄이고 성능을 향상시킴을 보였다. 실제 실험 결과, 제안한 윈도우 기반 알고리즘은 단순 접근법에 비해서 수배에서 수십 배까지 수행 시간을 줄인 것으로 나타났다.

제안한 알고리즘들은 스트림 시퀀스 매칭을 정확하게 수행할 수 있을 뿐 아니라, 실수 곱셈 연산의 최소화를 통하여 스트림의 길이, 선택률, 질의 시퀀스의 길이 등을 달리한 모든 경우에 있어서 단순 접근법에 비하여 성능을 크게 향상시킨 것으로 나타났다. 이 같은 결과를 볼 때, 제안한 스트림 시퀀스 매칭 알고리즘은 CPU 연산 능력이 부족한 핸드헬드 디바이스의 내장형 알고리즘으로 매우 적합하다고 사료된다.

참 고 문 헌

〔1〕 Agrawal, R., Faloutsos, C., and Swami, A.,

"Efficient Similarity Search in Sequence Databases," In *Proc. Int'l Conf. on FODO,* pp. 69-84, Oct. 1993.

〔2〕 Babcock, B., et al., "Models and Issues in Data Stream Systems," In *Proc. of ACM PODS*, pp. 1-16, June 2002.

〔3〕 Chan, K.-P., Fu, A. W.-C., and Yu, C. T., "Haar Wavelets for Efficient Similarity Search of Time-Series: With and Without Time Warping," *IEEE TKDE*, Vol.15, No.3, pp.686-705, Jan./Feb. 2003.

〔4〕 Faloutsos, C., Ranganathan, M., and Manolopoulos, Y., "Fast Subsequence Matching in Time-Series Databases," In *Proc. of ACM SIGMOD*, pp. 419-429, May 1994.

〔5〕 Fluke Electronics, http://www.fluke.com/.

〔6〕 Gao, L. and Wang, X. S., "Continually Evaluating Similarity-based Pattern Queries on a Stream Time Series," In *Proc. of ACM SIGMOD*, pp. 370-381, June 2002.

〔7〕 Gao, L., Yao, Z., and Wang, X. S., "Evaluating Continuous Nearest Neighbor Queries for Streaming Time Series via Pre-fetching," In *Proc. of ACM CIKM*, pp.485-492, 2002.

〔8〕 Medvidovic, N., et al., "Software Architectural Support for Handheld Computing," *IEEE Computer*, Vol.36, No.9, pp.66-73, Sept., 2003.

〔9〕 Moon, Y.-S., Whang, K.-Y., and Loh, W.-K., "Efficient Time-Series Subsequence Matching using Duality in Constructing Windows," *Information Systems*, Vol.26, No.4, pp.279-293, June, 2001.

〔10〕 Moon, Y.-S., Whang, K.-Y., and Han, W.-S., "General Match: A Subsequence Matching Method in Time-Series Databases Based on Generalized Windows," In *Proc. of ACM SIGMOD*, pp. 382-393, June 2002.

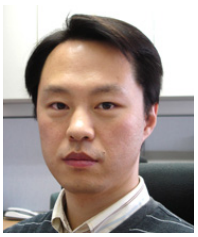
〔11〕 Stevens, W. R., Advanced Programming in the UNIX Environment, Addison-Wesley, 1992.

〔12〕 Wu, H., Salzberg, B., and Zhang, D., "Online Event-driven Subsequence Matching Over Financial Data Streams," In *Proc. of ACM SIGMOD*, pp. 23-34, June 2004.

문 양 세 (Yang-Sae Moon) 정회원

1991년 2월 한국과학기술원 과학기술대학 학사
1993년 2월 한국과학기술원 전산학과 석사
2001년 8월 한국과학기술원 전자전산학과 전산학전공 박사
1993년 2월~1997년 2월 현대전자산업(주) 주임연구원
2001년 9월~2002년 2월 (주)현대시스콤 선임연구원
2002년 2월~2005년 2월 (주)인프라밸리 기술위원(이사)
2005년 3월~현재 강원대학교 컴퓨터과학과 조교수
2005년 3월~현재 한국과학기술원 AITrc 연구원
<관심분야> 데이터 마이닝, 스트림 데이터, 저장 시스템, 데이터베이스 및 응용, 이동/무선 통신 서비스 및 시스템

노 웅 기 (Woong-Kee Loh) 정회원



1991년 2월 한국과학기술원 전산학과 학사
1993년 2월 한국과학기술원 전산학과 석사
2001년 2월 한국과학기술원 전산학과 박사
2001년 2월~2003년 9월 (주)티맥스소프트 책임연구원
2003년10월~2005년 3월 (주)티맥스데이타 수석연구원
2005년 4월~현재 한국과학기술원 초빙교수
<관심분야> 데이터 마이닝/데이터 웨어하우징, 스트림 데이터 마이닝, 웹 마이닝, 정보 검색, 멀티미디어 데이터베이스, 멀티미디어 내용기반 검색, 공간 데이터베이스, 임베디드 데이터베이스, 데이터베이스 시스템 엔진