시계열 서브시퀀스 매칭에서 발생하는 성능 병목의 효과적인 해결 방안

김 상욱*, 오 세봉**

한양대학교 정보통신대학*, 티맥스 소프트(주)**

Effective Resolving of the Performance Bottleneck in Time-Series Subsequence Matching

Sang-Wook Kim* and Se-Bong Oh**

College of Information and Communications in Kangwon National University*, TmaxSoft Inc.**

요약문

서브시퀀스 매칭은 주어진 질의 시퀀스와 변화의 추세가 유사한 서브시퀀스들을 시계열 데이터베이스로부터 검색하는 연산이다. 본 논문에서는 서브시퀀스 매칭 처리의 성능 병목을 파악하고, 이를 해결함으로써 전체 서브시퀀스 매칭의 성능을 크게 개선하는 방안에 관하여 논의한다. 먼저, 사전 실험을 통하여 후처리 단계가 서브시퀀스 매칭의 성능 병목이며, 후처리 단계의 최적화가 기존의 서브시퀀스 매칭 기법들이 간과한 매우 중요한 이슈임을 지적한다. 이러한 서브시퀀스 매칭의 성능 병목을 해결하기 위하여 후처리 단계를 최적으로 처리할 수 있는 간단하면서도 매우 효과적인 기법을 제안한다. 제안된 기법은 후처리 단계에서 후보 서브시퀀스들이 질의 시퀀스와 실제로 유사한가를 판단하는 순서를 조정함으로써 기존의 후처리 단계의 처리에서 발생하는 많은 디스크 액세스의 중복과 CPU 처리의 중복을 완전히 제거할 수 있다. 실제 데이터와 생성 데이터를 이용한 다양한 실험들을 통하여 제안된 기법의 성능 개선 효과를 정량적으로 검증한다.

1. 서론

시계열 데이터베이스(time-series database)란 객체의 변화되는 값들의 연속으로 구성된 데이터 시퀀스(data sequence)들의 집합이다[Agr93]. 유사 시퀀스 매칭(similar sequence matching)이란 주어진 질의 시퀀스(query sequence)와 변화의 패턴이 유사한 시퀀스들을 시계열 데이터베이스로부터 찾아내는 연산이다[Agr93][Fal94][Moo01]. 길이가 동일한 서로 다른 두 시퀀스 X(=<x0, x1, ... , xn-1>)와 Y(=<y0, y1, ... , yn-1>)간의 유사성을 측정하는 척도로서 아래와 같이 정의되는 유클리드 거리(Euclidean distance) D(X, Y)를 널리 사용한다[Agr93][Fal94][Kim01][Loh00][Par01]. D(X, Y)가 ε이하인 임의의 두 시퀀스 X, Y는 ε-매치(ε-match)한다고 한다[Moo01].

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

유사 시퀀스 매칭은 다음과 같이 전체 매칭(whole matching)과 서브시퀀스 매칭(subsequence matching)으로 구분된다[Fal94].

- 전체 매칭: 데이터베이스 D 내에 존재하는 시퀀스 S1, S2, ..., SN에 대하여 질의 시퀀스 Q와 허용치 ε이 주어질 때, D로부터 Q와 ε-매치하는 시퀀스 Si를 검색한다.
- 서브시퀀스 매칭: 데이터베이스 D 내에 존재하는 시퀀스 S1, S2, ..., SN에 대하여 질의 시퀀스 Q와 허용치 ε이 주어질 때, D로부터 Q와 ε-매치하는 서브시퀀스 X를 포함하는 시퀀스 Si와 이 서브시퀀스 X가 Si내에서 시작하는 위치를 검색한다.

참고 문헌 [Fal94] 및 [Moo01]에서는 서브시퀀스 매칭을 위한 처리 기법을 제안하였다. 참고 문헌 [Moo01]의 명칭을 따라 본 논문에서는 [Fal94]의 기법을 FRM, [Moo01]의 기법을 Dual-Match라 부른다. 두 기법 모두 미리 지정된 고정된 길이 w를 갖는 윈도우(window) 개념을 이용한다. 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*-트리에 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.

본 논문에서는 FRM과 Dual-Match 모두에서 후처리 단계가 서브시퀀스 매칭의 성능 병목이며, 후처리 단계의 최적화가 기존의 서브시퀀스 매칭 기법들이 간과한 매우 중요한 이슈임을 지적한다. 이러한 서브시퀀스 매칭의 성능 병목을 해결하기 위하여 후처리 단계를 최적으로 처리할 수 있는 간단하면서도 매우 효과적인 기법을 제안한다. 제안된 기법은 후처리 단계에서 후보 서브시퀀스들이 질의 시퀀스와 실제로 유사한가를 판단하는 순서를 조정함으로써 기존의 후처리 단계의 처리에서 발생하는 많은 디스크 액세스의 중복과 CPU 처리의 중복을 완전히 제거할 수 있다. 실제 데이터와 생성 데이터를 이용한 다양한 실험들을 통하여 제안된 기법의 성능 개선 효과를 정량적으로 검증한다.

2. 서브시퀀스 매칭의 성능 병목

2.1. 사전 실험 결과 및 분석

본 연구에서는 먼저 위에서 서브시퀀스 매칭의 전체 처리 시간 중 각 요소가 차지하는 비중을 분석하기 위하여 다음과 같은 사전 실험을 수행하였다. 사용된 실험 데이터는 길이가 1,024인 620개의 한국의 실제 주식 데이터이다. 저차원 변환을 위해서는 DFT를 사용하고, 각 시퀀스로부터 인덱싱을 위한 6개의 특성을 추출함으로써 6차원 R*-트리를 구성하였다. 사용된 질의 시퀀스의 길이는 768이며, 허용치 ε은 15개의 최종 질의 결과가 나오도록 설정하였다. 윈도우의 길이로는 FRM의 경우 512, Dual-Match의 경우 256으로 설정하였다. 그림 2.1과 그림 2.2는 각각 FRM 및 Dual-Match에 대한 위의 사전 실험 결과를 나타낸 것이다. 가로축은 서브시퀀스 매칭의 처리에 소요되는 각 요소를 나타내며, 세로축은 각 요소에서 소요된 시간을 나타낸다.

먼저, 그림 2.1에 나타난 FRM의 결과를 보면, 전체 처리 시간 중 후처

리 단계에서 소요된 시간(PP-Time(CPU)+PP-Time(DISK))이 85%를 차지한다. 특히, PP-Time(DISK)가 전체 처리 시간의 50%에 가까운 비중을 보였다. 그림 2.2에 나타난 Dual-Match의 결과에서는 전체 처리 시간 중 후처리 단계에서 소요된 시간(PP-Time(CPU)+PP-Time(DISK))이 82%를 차지한다. PP-Time(DISK)는 전체 처리 시간의 77%에 가까운 비중을 보였다. 또한, Dual-Match는 점 역과 효과[Moo01]를 이용함으로써 FRM과 비교하여 전체 처리 시간을 약 50%로 감소시킬 수 있었다.

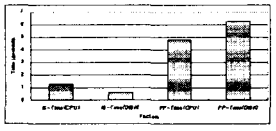

그림 2.1. FRM의 소요 시간.

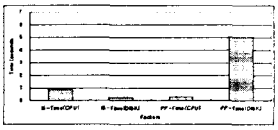

그림 2.2. Dual-Match의 소요 시간.

2.2. 서브시퀀스 매칭의 성능 병목 및 해결 방안

사전 실험 결과를 보면, 서브시퀀스 매칭의 처리 시간 중 후처리 단계에서 소요되는 시간은 FRM과 Dual-Match에서 각각 85%, 77%로서 이것은 서브시퀀스 매칭의 성능 병목이 후처리 단계에 있음을 의미한다. 또한, 후처리 단계에서 소요되는 시간을 줄임으로써 전체 서브시퀀스 매칭의 성능을 크게 개선할 수 있음을 의미하는 것이다.

후처리 단계에서는 인덱스 검색 단계에서 반환되는 후보 서브시퀀스들을 디스크로부터 액세스하여 질의 시퀀스와의 실제 유클리드 거리를 계산한다. PP-Time(DISK)와 PP-Time(CPU)는 착오 채택의 수에 비례하므로 후처리 단계에서 소요되는 시간을 줄일 수 있는 중요한 전략의 하나는 착오 채택의 수를 줄이는 것이다. FRM 혹은 Dual-Match에서 착오 채택의 수를 추가로 줄이기 위하여 높은 (1) R*-트리 차원의 사용, (2) 좋은 변환 함수의 사용, (3) 큰 윈도우의 사용 등의 방법들을 고려할 수 있다.

그러나, 위의 모든 방법을 최대한 동원하더라도 결국 착오 채택의 발생은 불가피하며, 실제로 발생하는 착오 채택의 수는 최종 질의 결과 수와 비교하여 매우 크다. 착오 채택의 수를 줄이기 위한 좋은 기법으로 평가받는 Dual-Match도 전술한 21개의 최종 질의 결과를 가지는 사전 실험에서 30,781개라는 엄청난 수의 착오 채택을 유발시킨 것을 참고할 필요가 있다.

본 연구에서는 후처리 단계에서 소요되는 시간을 줄일 수 있는 또 하나의 중요한 전략으로서 ***후처리 단계의 수행 방식*** 자체를 개선한 것을 제시한다. 이 전략은 동일한 수의 후보 서브시퀀스들이 반환된 경우라도, 후처리 단계가 효과적으로 수행된다면 처리 성능을 개선시킬 수 있다는 점에서 착안한 것이다. 기존의 기법들[Fal94][Moo01] 등은 주로 위와 같이 착오 채택의 수를 줄이는 방안에 관해서만 연구의 초점을 맞추었으며, 좋은 후처리 단계의 처리 방식을 고안하는 전략은 간과하였다. 본 논문에서는 좋은 후처리 단계의 수행 방식을 제시하고, 이를 통하여 전체 서브시퀀스 매칭의 성능을 크게 향상시키고자 한다.

3. 제안하는 기법

3.1. 기존 기법들의 문제점

그림 3.1은 인덱스 검색 단계와 후처리 단계로 구성되는 FRM과 Dual-Match의 공통적인 처리 과정을 도면화 한 것이다. 그림 3.1에서 점으로 채워진 네 개의 삼각형은 인덱스 검색 단계에서 범위 질의에 의하여 액세스되는 R*-트리의 일부분을 나타낸다. 각 범위 질의의 결과는 그 범위 질의에서 사용된 질의 윈도우 점과의 유클리드 거리가 $\varepsilon/\sqrt{p}$ 이하인 데이터 윈도우 점들의 집합이다. 후처리 단계에서는 먼저 인덱스 검색에서 반환되는 이러한 각 후보 윈도우에 대하여 그 윈도우가 속하는 후보 서브시퀀스를 파악한다. 이 후보 서브시퀀스가 포함되는 데이터 시퀀스를 디스크로부터 액세스함으로써 이 후보 서브시퀀스의 착오 채택 여부를 확인한다. 그림 3.1에서 각 삼각형과 실선으로 연결된 아래쪽의 각 사각형은 이러한 후보 서브시퀀스를 포함하는 데이터 시퀀스를 나타내며, 이들은 후처리 단계에서 디스크로부터 액세스된다.

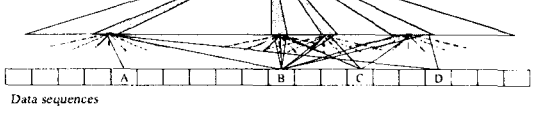

그림 3.1. 서브시퀀스 매칭의 처리 과정.

R*-트리 내에 저장된 데이터 윈도우 점들은 동일한 시퀀스로부터 추출되었는지의 여부에 관계없이 모두 독립적으로 관리된다. 또한, 기존의 FRM 및 Dual-Match의 후처리 단계에서는 인덱스 검색을 통하여 반환되는 순서로 각 후보 윈도우가 포함되는 후보 서브시퀀스를 질의 시퀀스와 비교한다[Fal94][Moo01]. 이러한 처리 방식은 다음과 같은 두 가지 측면에서 성능상의 문제들을 야기 시킨다.

(1) 디스크 액세스 오버헤드

이것은 동일한 데이터 시퀀스를 디스크로부터 반복적으로 액세스함으로써 발생하는 성능상의 문제이다. 이 문제는 인덱스 검색의 결과, 같은 데이터 시퀀스에 속하는 서로 다른 윈도우들이 후보 윈도우로 선택되는 경우에 발생한다. 즉, 같은 시퀀스에 속하는 후보 윈도우들이라 할 지라도 인덱스 검색의 결과로 반환되는 시점은 서로 다르므로 각각의 처리를 위해 같은 데이터 시퀀스를 디스크로부터 여러 번 액세스해야 하는 것이다.

(2) CPU 처리 오버헤드

이것은 동일한 서브시퀀스를 질의 시퀀스와 두 번 이상 비교함으로써 발생하는 CPU 성능상의 문제이다. 이 문제는 인덱스 검색의 결과, 동일한 데이터 서브시퀀스에 속하는 서로 다른 윈도우들이 후보 윈도우로 선택되는 경우에 발생한다. 이러한 불필요한 중복 확인은 서브시퀀스 매칭을 위한 PP-Time(CPU)의 증가를 초래한다.

3.2. 해결 방안

디스크 액세스 및 CPU 처리의 중복 문제가 발생하는 근본적인 원인은 (1) 데이터 윈도우들이 동일한 시퀀스 혹은 서브시퀀스로부터 추출되었는지의 여부에 관계없이 R*-트리 내에 독립적으로 저장되며, (2) 후처리 단계에서 인덱스 검색의 결과로 반환되는 순서 그대로 후보 서브시퀀스의 착오 채택 여부를 검사한다는 데에 있다. 본 연구에서는 이러한 문제를 해결하기 위한 간단하면서도 매우 효과적인 ***윈도우 순서 기법***을 제안한다. 제안하는 기법의 기본 전략은 인덱스 검색의 결과로 반환되는 후보 윈도우들의 처리 순서를 조정함으로써 동일한 시퀀스에 속하는 후보 윈도우들, 같은 서브시퀀스에 속하는 후보 윈도우들을 연속적으로 처리한다는 것이다.

이를 위한 구체적인 방법은 알고리즘 3.1과 같다. 먼저, 단계 1에서는 각 후보 윈도우와 대응되는 후보 서브시퀀스의 식별자 <seqID, subseqOffset>를 인덱스 검색 단계에 의하여 반환되는 순서대로 주기억장치 내에 저장한다. 여기서 seqID는 이 후보 윈도우가 속하는 데이터 시퀀스의 식별자이다. 또한, subseqOffset은 seqID가 의미하는 시퀀스 내의 이 후보 윈도우와 대응되는 후보 서브시퀀스의 시작 위치(offset)이다.

단계 2에서는 <seqID, subseqOffset>를 정렬 키(sort key)로 사용하여 단계 1에서 저장한 모든 후보 서브시퀀스 식별자들을 정렬한다. 단계 3에서는 각 후보 서브시퀀스를 단계 2에 의하여 정렬된 순서로 액세스하고, 유클리드 거리를 계산함으로써 이것이 질의 시퀀스와 ε-매치 하는가를 실제로 확인한다. 이 과정에서 해당 후보 서브시퀀스 식별자가 직전 루프에서 처리된 것과 동일한 경우에는 이러한 착오 채택 여부의 확인 작업을 수행하지 않는다. 또한, 해당 후보 서브시퀀스가 속하는 시퀀스가 직전 루프에서 처리되지 않은 경우에 한하여 이 시퀀스를 디스크로부터 읽어들인다.

1. **FOR** each *candidate window* returned by the index search step,
   store the identifier <seqID, subseqOffset> of the *candidate subsequence* corresponding to the candidate window;
2. Sort the identifiers of candidate subsequences by using <seqID, subseqOffset> as a sort key;
3. **FOR** each identifier in the sorted order,
   3.1. **IF** it is exactly the same as the one processed in the previous loop, skip it and go to the next loop;
   3.2. **ELSE**
      IF the seqID is different from the one in the previous loop, access the corresponding subsequence from disk;
      IF the candidate subsequence is actually ε-matched with the query sequence, insert it in the final answer

알고리즘 3.1. 윈도우 순서 기법을 이용한 후처리.

4. 성능 평가

4.1. 실험 환경

본 연구에서는 성능 분석을 위하여 실제 데이터베이스 K_Stock_Data와 합성 데이터 Syn_Data를 사용하였다. K_Stock_Data는 사전 실험에서 사용하였던 것과 동일하며, 합성 데이터 Syn_Data내의 각 시퀀스 S = <s1, s2, ..., sn> 는 다음과 같은 랜덤 워크(random walk) 형태를 가진다

[Agr93].

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

여기서 zi는 구간 [-0.1, 0.1] 사이에서 균일한 분포를 취하는 랜덤 변수이며, 시퀀스의 첫 요소 값 s1은 구간 [1, 10] 사이의 임의의 값을 취하도록 하였다. 질의 시퀀스 Q는 데이터베이스로부터 선택한 시퀀스로부터 길이가 Len(Q)인 임의의 서브시퀀스를 선택하였다. 성능 평가를 위한 하드웨어 플랫폼 및 소프트웨어 플랫폼 등의 실험 환경은 사전 실험과 동일하다.

제안된 윈도우 순서 기법의 성능 비교 대상은 기존의 인덱스 검색 단계에서 반환하는 순서대로 후처리를 수행하는 기존의 기법이다. 참고 문헌 [Moo01]에서 이미 Dual-Match가 FRM보다 월등하게 우수한 성능을 가지는 것으로 판명되었으므로, 본 성능 평가에서는 제안된 기법과 기존의 기법을 Dual-Match에 적용함으로써 두 기법의 성능을 비교하였다.

4.2. 실험 결과

먼저, 실험 1에서는 K_Stock_Data로부터 추출한 길이 256의 디스조인트 윈도우들을 포함하는 R*-트리에 대하여 제안된 기법과 기존 기법을 채택하는 서로 다른 두 가지 Dual-Match 기반 서브시퀀스 매칭 수행하였다. 사용된 각 질의 시퀀스의 길이는 512이다.

표 4.1은 실험 결과를 나타낸 것이다. 먼저, 선택률(selectivity)[Sel79]이 10^{-4}인 경우의 PP-Time(DISK)와 PP-Time(CPU)를 분석한다. PP-Time(DISK)의 경우, 제안된 기법이 기존 기법과 비교하여 30(=5.896/0.195) 배 이상의 성능 개선 효과를 가지는 것으로 나타났다. 표 4.1에 나타난 바와 같이, 제안된 기법은 실제로 디스크로부터 액세스되는 시퀀스들의 수(# of sequences accessed)를 기존 기법의 약 1/3(=333/938)로 줄였다. 그러나 각 시퀀스를 디스크로부터 랜덤하게 액세스하는 기존의 기법과는 달리, 제안된 기법은 시퀀스 식별자 순서대로 저장된 시퀀스들을 디스크로부터 연속적으로 액세스하는 특성을 갖는다. 이러한 특성으로 제안된 기법은 이 외에도 디스크의 탐색 시간(seek time)을 크게 줄일 수 있으며, 이것이 성능 개선 효과에 반영된 것이다 [Web98].

표 4.1. 선택률의 변화에 따르는 제안된 기법의 성능 개선 효과.

selectivity	# of candidates compared		# of sequences accessed		PP-Time(CPU) (unit: second)		PP-Time(DISK) (unit: second)		Total (unit: second)	
	our method	prev. method	our method	prev. method	our method	prev. method	our method	prev. method	our method	prev. method
10^{-4}	81,245	96,189	333	938	0.733	0.780	0.195	5.896	1.473	7.232
5×10^{-4}	125,710	155,038	364	1,247	1.118	1.228	0.191	7.758	1.914	9.600
10^{-3}	143,404	180,871	374	1,338	1.276	1.367	0.184	8.335	2.090	10.330

PP-Time(CPU)의 경우, 후보 서브시퀀스 식별자들을 정렬해야 하는 추가의 오버헤드가 있음에도 제안된 기법이 기존 기법과 비교하여 더 나은 성능을 보였다. 이것은 동일한 후보 서브시퀀스를 질의 시퀀스와 중복하여 비교하는 기존의 기법의 문제점을 제안된 기법이 해결하였기 때문이다. 이것은 제안된 기법에서 요구하는 정렬 오버헤드가 전체 성능에 부정적인 영향을 미치지 않음을 보여주는 것이다.

전체 후처리 단계 수행 시간(PP-Time(CPU)+PP-Time(DISK))의 경우, 제안된 기법은 기존 기법의 성능을 7.19(=(0.780+5.896)/(0.733+0.195)) 배 개선한 것으로 나타났다. 또한, 전체 서브시퀀스 매칭의 수행 시간(Total)의 경우, 제안된 기법을 후처리 단계에 적용함으로써 얻게 되는 성능 개선 효과는 4.91(=7.232/1.473) 배로 나타났다. 선택률 5×10^{-4} 및 10^{-3}에 대한 실험 결과도 선택률 10^{-4}인 경우의 실험 결과와 매우 유사한 경향을 보였다.

K_Stock_Data는 비교적 소규모의 데이터베이스이다. 본 성능 평가에서는 제안된 기법이 대규모 데이터베이스에서도 효과적으로 동작함을 규명하기 위하여 다양한 시퀀스의 수를 갖도록 생성한 대규모 Syn_Data를 이용한 실험을 추가로 수행하였다.

실험 2에서는 데이터 시퀀스의 길이는 1,000으로 고정시킨 상태에서 데이터 시퀀스들의 수를 5,000, 10,000, 15,000, 20,000, 25,000으로 변화하면서 제안된 기법과 기존 기법을 채택하는 서로 다른 두 가지 Dual-Match 기반 서브시퀀스 매칭의 성능을 비교하였다. 사용된 질의 시퀀스의 길이는 500이며, 질의 선택률은 10^{-3}이다. 또한, 인덱싱을 위하여 사용된 윈도우의 길이는 250이다.

그림 4.1은 실험 결과를 나타낸 것이다. 가로축은 데이터 시퀀스의 수를 나타내며, 세로축은 서브시퀀스 매칭을 위한 후처리 단계 수행 시간(PP-Total) 및 전체 수행 시간(Total)을 나타낸다. 데이터 시퀀스의 수가 증가함에 따라 두 가지 기법을 위한 후처리 단계 수행 시간은 거의 선형적으로 증가하는 것으로 나타났다. 후처리 단계 수행 시간의 증가율에서는 두 기법이 큰 차이를 보였다. 이것은 제안된 기법을 채택하는 경우, 후처리 과정에서 나타나는 시퀀스 액세스의 중복과 서브시퀀스 비교의 중복을 완전히 제거할 수 있기 때문이다. 후처리 단계 수행 시간에 대한 제안된 기법의 성능 개선 효과는 약 5.14 배에서 약 5.22 배 사이로 나타났다.

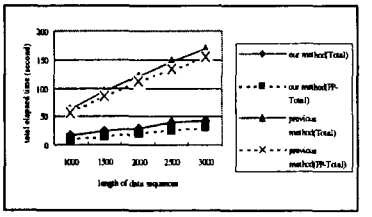

그림 4.1. 데이터 시퀀스 수의 변화에 따르는 제안된 기법의 성능 개선 효과.

그림 4.1에서 기존 기법을 사용하는 후처리 단계의 수행 시간은 모든 경우에서 전체 수행 시간의 90% 이상인 반면, 제안된 기법을 사용하는 후처리 단계의 수행 시간은 모든 경우에서 전체 수행 시간의 70% 이하인 것으로 나타났다. 이것은 제안된 기법이 서브시퀀스 매칭의 성능 병목을 성공적으로 해결하였음을 보여주는 것이다. 제안된 기법을 통한 성능 병목의 해결 결과, 전체 수행 시간은 약 3.88 배에서 4.21배 개선되는 것으로 나타났다.

5. 결론

본 논문에서는 기존의 서브시퀀스 매칭 처리의 성능 병목을 파악하고, 이를 신속하게 처리하기 위한 간단하면서도 매우 효과적인 윈도우 순서 기법을 제안하였다. 실험 결과에 의하면, 제안된 기법을 채택함으로써 전체 서브시퀀스 매칭에 소요되는 시간의 90%에 이르던 후처리 단계의 비중을 70%이하로 내릴 수 있었다. 이것은 제안된 기법이 서브시퀀스 매칭의 성능 병목을 성공적으로 해결하였음을 시사하는 것이다. 이 결과, 제안된 기법은 전체 서브시퀀스 매칭의 성능을 5.60 배까지 향상시키는 효과를 보였다. 현재, 제안된 기법을 다양한 변환을 지원하는 서브시퀀스 매칭에 적용하는 방안과 그 성능 개선 효과를 정량적으로 규명하는 문제에 대하여 연구하고 있다.

6. 참고 문헌

[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.

[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.

[Kim01] S. W. Kim, S. H. Park, and W. W. Chu, "An Index-Based Approach for Similarity Search Supporting Time Warping in Large Sequence Databases," In *Proc. Int'l. Conf. on Data Engineering*, IEEE ICDE, pp. 607-614, 2001.

[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.

[Par01] S. H. Park, S. W. Kim, I. S. Cho, and S. Padmanabhan, "Prefix-Querying: An Approach for Effective Subsequence Matching Under Time Warping in Sequence Databases," In *Proc. ACM Intl. Conf. on Information and Knowledge Management*, ACM CIKM, pp. 255-262, 2001.

[Sel79] P. G. Selinger et al., "Access Path Selection in a Relational Database Management System," In *Proc. Int'l. Conf. on Management of Data*, ACM SIGMOD, pp.23-34, May 1979.

[Web98] R. Weber, H.-J. Schek, and S. Blott, "A Quantitative Analysis and Performance Study for Similarity Search Methods in High-Dimensional Spaces," In *Proc. Int'l. Conf. on Very Large Data Bases*, VLDB, pp. 194-205, 1998.